AIX Version 6.1

Commands Reference, Volume 2, d - h

SC23-5244-00

AIX Version 6.1

Commands Reference, Volume 2, d - h

SC23-5244-00

Note
Before using this information and the product it supports, read the information in “Notices,” on page 715.

First Edition (November 2007)

This edition applies to AIX Version 6.1 and to all subsequent releases of this product until otherwise indicated in new editions.

A reader’s comment form is provided at the back of this publication. If the form has been removed, address comments to Information Development, Department 04XA-905-6C006, 11501 Burnet Road, Austin, Texas 78758-3493. To send comments electronically, use this commercial Internet address: aix6kpub@austin.ibm.com. Any information that you supply may be used without incurring any obligation to you.

© Copyright International Business Machines Corporation 1997, 2007. All rights reserved.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

About This Book ix
How to Use This Book ix
ISO 9000 xi
32-Bit and 64-Bit Support for the Single UNIX Specification xii
Related Information xii

Alphabetical Listing of Commands 1
dacinet Command 1
dadmin Command 2
date Command 4
dbts Command 8
dbx Command 9
dc Command 66
dd Command 68
defaultbrowser Command 72
defif Method 73
definet Method 74
defragfs Command 75
defvsd Command 78
deleteX11input Command 80
delta Command 81
deroff Command 83
detachrset Command 84
devinstall Command 85
devnm Command 87
df Command 87
dfmounts Command 92
dfpd Command 93
dfsck Command 94
dfshares Command 96
dhcpaction Command 97
dhcpcd Daemon 98
dhcpcd6 Daemon 100
dhcprd Daemon 102
dhcpsconf Command 103
dhcpsd Daemon 105
dhcpsdv6 Daemon 106
diag Command 108
diaggetrto Command 111
diagrpt Command 112
diagsetrto Command 113
diction Command 114
diff Command 115
diff3 Command 118
diffmk Command 120
dig Command 121
digest Command 125
dircmp Command 126
dirname Command 127
disable Command 128
diskusg Command 130
dispgid Command 132
dispuid Command 133

© Copyright IBM Corp. 1997, 2007

dist Command 134
dmadm Command. 137
dmf Command 138
dmpuncompress Command 167
dms Command 168
dms_enable_fs Command 171
dnssec-keygen Command 171
dnssec-makekeyset command 173
dnssec-signkey Command. 174
dnssec-signzone Command 175
dodisk Command 176
domainname Command 178
dosdel Command 178
dosdir Command 179
dosformat Command. 181
dosread Command 183
doswrite Command 184
dp Command 186
dpid2 Daemon 186
drm_admin Command 188
drmgr Command 191
drslot Command 193
dscreen Command 195
dslpaccept Command 196
dslpaccess Command 197
dslpadmin Command. 198
dslpdisable Command 202
dslpenable Command 203
dslpprotocol Command 203
dslpreject Command 205
dslpsearch Command 206
dspcat Command 207
dspmsg Command 208
dtaction Command 210
dtappintegrate Command 212
dtlogin Command 213
dtscript Command. 239
dtsession Command 240
dtterm Command 248
du Command 257
dump Command 259
dumpcheck Command 261
dumpctrl Command 262
dumpfs Command. 267
echo Command 268
ed or red Command 270
edit Command 302
edquota Command 309
efsenable Command 311
efskeymgr Command 313
efsmgr Command 317
egrep Command 319
eimadmin Command 321
elogevent Command 329
emgr Command 331
emstat Command 335

emsvcsctrl Command . . . 337
enable Command . . . 340
enotifyevent Command . . . 342
enq Command . . . 343
enroll Command . . . 352
enscript Command . . . 352
entstat Command . . . 358
env Command . . . 363
epkg Command . . . 365
eqn Command . . . 372
errclear Command . . . 374
errctrl Command . . . 376
errdead Command . . . 381
errdemon Daemon . . . 382
errinstall Command . . . 384
errlogger Command . . . 387
errmsg Command . . . 388
errpt Command . . . 390
errstop Command . . . 396
errupdate Command . . . 396
ethchan_config Command . . . 404
ewallevent Command . . . 405
ex Command . . . 407
execerror Command . . . 408
execrset Command . . . 409
expand Command . . . 410
expfilt Command . . . 412
explain Command . . . 413
explore Command . . . 413
exportfs Command . . . 414
exportvg Command . . . 422
expr Command . . . 423
exptun Command . . . 427
extendlv Command . . . 428
extendvg Command . . . 430
f Command . . . 432
factor Command . . . 434
fc Command . . . 435
fccheck Command . . . 438
fcclear Command . . . 439
fcdecode Command . . . 442
fcdispfid Command . . . 443
fcfilter Command . . . 444
fcinit Command . . . 446
fclogerr Command . . . 449
fcpushstk Command . . . 456
fcreport Command . . . 462
fcstat Command . . . 463
fcstkrpt Command . . . 466
fcteststk Command . . . 468
fddistat Command . . . 470
fdformat Command . . . 472
fdpr Command . . . 473
fencevsd Command . . . 479
ff Command . . . 481
fg Command . . . 483

fgrep Command . . . 484
file Command . . . 486
filemon Command . . . 488
fileplace Command . . . 496
find Command . . . 498
finger Command . . . 506
fingerd Daemon . . . 509
fish Command . . . 510
flcopy Command . . . 511
flush-secldapclntd Command . . . 512
fmt Command . . . 513
fold Command . . . 514
folder Command . . . 515
folders Command . . . 518
format Command . . . 520
fortune Command . . . 522
forw Command . . . 523
fpm Command . . . 526
frcactrl Command . . . 530
from Command . . . 532
fsck Command . . . 533
fsck_cachefs Command . . . 537
fsdb Command . . . 538
fsplit Command . . . 549
ftp Command . . . 550
ftpd Daemon . . . 564
fuser Command . . . 571
fwtmp Command . . . 573
fxfer Command . . . 574
gated Daemon . . . 586
gdc Command . . . 589
gencat Command . . . 591
gencopy Command . . . 592
gencore Command . . . 594
genfilt Command . . . 594
geninstall Command . . . 597
genkex Command . . . 599
genkld Command . . . 599
genld Command . . . 600
gennames Command . . . 601
gensyms Command . . . 601
gentun Command . . . 602
genxlt Command . . . 605
get Command . . . 607
getconf Command . . . 616
getdev Command . . . 623
getdgrp Command . . . 625
getea Command . . . 628
getopt Command . . . 629
getopts Command . . . 630
getrunmode Command . . . 632
getsecconf Command . . . 633
getsyslab Command . . . 634
gettable Command . . . 635
gettrc Command . . . 635
getty Command . . . 636

glbd Daemon 638
gprof Command 641
grap Command 646
greek Command 650
grep Command 650
groups Command 653
grpck Command 654
grpsvcsctrl Command 657
gssd Daemon 660
ha_star Command. 660
ha.vsd Command 661
ha_vsd Command 665
haemd Daemon 666
haemd_HACMP Command 666
haemqvar Command 667
haemtrcoff Command 671
haemtrcon Command 673
haemunlkrm Command 675
hagsd Daemon 677
hagsns Command 679
hagsvote Command 681
halt or fasthalt Command 683
hangman Command 685
hatsoptions Command 686
hash Command 687
head Command 688
help Command 689
host Command 690
hostent Command 692
hostid Command 694
hostmibd Daemon 695
hostname Command 697
hosts2ldif Command 698
hp Command 699
hplj Command 699
hpmcount Command 701
hpmstat Command 706
hps_dump Command 710
htable Command 711
hty_load Command 713
hyphen Command. 714

Appendix. Notices 715
Trademarks 716

Index 719

About This Book

This book provides end users with complete detailed information about commands for the AIX operating system. The commands are listed alphabetically and by category, and complete descriptions are given for commands and their available flags. If applicable, each command listing contains examples. This volume contains AIX commands that begin with the letters d through h. This publication is also available on the documentation CD that is shipped with the operating system.

How to Use This Book

A command is a request to perform an operation or run a program. You use commands to tell the operating system what task you want it to perform. When commands are entered, they are deciphered by a command interpreter (also known as a shell) and that task is processed.

Some commands can be entered simply by typing one word. It is also possible to combine commands so that the output from one command becomes the input for another command. This is known as pipelining.

Flags further define the actions of commands. A flag is a modifier used with the command name on the command line, usually preceded by a hyphen.

Commands can also be grouped together and stored in a file. These are known as shell procedures or shell scripts. Instead of executing the commands individually, you execute the file that contains the commands.

Some commands can be constructed using Web-based System Manager applications or the System Management Interface Tool (SMIT).

Highlighting

The following highlighting conventions are used in this book:

Bold	Identifies commands, subroutines, keywords, files, structures, directories, and other items whose names are predefined by the system. Also identifies graphical objects such as buttons, labels, and icons that the user selects.
Italics	Identifies parameters whose actual names or values are to be supplied by the user.
`Monospace`	Identifies examples of specific data values, examples of text similar to what you might see displayed, examples of portions of program code similar to what you might write as a programmer, messages from the system, or information you should actually type.

Format

Each command may include any of the following sections:

Purpose	A description of the major function of each command.
Syntax	A syntax statement showing command line options.
Description	A discussion of the command describing in detail its function and use.
Flags	A list of command line flags and associated variables with an explanation of how the flags modify the action of the command.
Parameters	A list of command line parameters and their descriptions.
Subcommands	A list of subcommands (for interactive commands) that explains their use.
Exit Status	A description of the exit values the command returns.
Security	Specifies any permissions needed to run the command.
Examples	Specific examples of how you can use the command.
Files	A list of files used by the command.
Related Information	A list of related commands in this book and related discussions in other books.

© Copyright IBM Corp. 1997, 2007

Reading Syntax Statements

Syntax statements are a way to represent command syntax and consist of symbols such as brackets ([ ]), braces ({ }), and vertical bars (|). The following is a sample of a syntax statement for the **unget** command:

unget [-**r***SID*] [-**s**] [-**n**] *File* ...

The following conventions are used in the command syntax statements:

- Items that must be entered literally on the command line are in **bold**. These items include the command name, flags, and literal charactors.
- Items representing variables that must be replaced by a name are in *italics*. These items include parameters that follow flags and parameters that the command reads, such as *Files* and *Directories*.
- Parameters enclosed in brackets are optional.
- Parameters enclosed in braces are required.
- Parameters not enclosed in either brackets or braces are required.
- A vertical bar signifies that you choose only one parameter. For example, [a | b] indicates that you *can* choose a, b, or nothing. Similarly, { a | b } indicates that you *must* choose either a or b.
- Ellipses (...) signify the parameter can be repeated on the command line.
- The dash (-) represents standard input.

Listing of Installable Software Packages

To list the installable software package (fileset) of an individual command, use the **lslpp** command with the **-w** flag. For example, to list the fileset that owns the **installp** command, enter:

```
lslpp -w /usr/sbin/installp
```

Output similar to the following displays:

```
File                            Fileset                 Type
-----------------------------------------------------------------------
/usr/sbin/installp                bos.rte.install           File
```

To list the fileset that owns all file names that contain `installp`, enter:

```
lslpp -w "*installp*"
```

Output similar to the following displays:

```
File                            Fileset                 Type
-----------------------------------------------------------------------
/usr/sbin/installp                bos.rte.install           File
/usr/clvm/sbin/linstallpv         prpq.clvm                 File
/usr/lpp/bos.sysmgt/nim/methods/c_installp
                                  bos.sysmgt.nim.client     File
```

Running Commands in the Background

If you are going to run a command that takes a long time to process, you can specify that the command run in the background. Background processing is a useful way to run programs that process slowly. To run a command in the background, you use the & operator at the end of the command:

```
Command&
```

Once the process is running in the background, you can continue to work and enter other commands on your system.

At times, you might want to run a command at a specified time or on a specific date. Using the **cron** daemon, you can schedule commands to run automatically. Or, using the **at** and **batch** commands, you can run commands at a later time or when the system load level permits.

Entering Commands

When you work with the operating system, you typically enter commands following the shell prompt on the command line. The shell prompt can vary. In the following examples, $ is the prompt.

To display a list of the contents of your current directory, you would type `ls` and press the Enter key:

```
$ ls
```

When you enter a command and it is running, the operating system does not display the shell prompt. When the command completes its action, the system displays the prompt again. This indicates that you can enter another command.

The general format for entering operating system commands is:

Command Flag(s) Parameter

The flag alters the way a command works. Many commands have several flags. For example, if you type the **-l** (long) flag following the **ls** command, the system provides additional information about the contents of the current directory. The following example shows how to use the **-l** flag with the **ls** command:

```
$ ls -l
```

A parameter consists of a string of characters that follows a command or a flag. It specifies data, such as the name of a file or directory, or values. In the following example, the directory named **/usr/bin** is a parameter:

```
$ ls -l /usr/bin
```

When entering commands in, it is important to remember the following:

- Commands are usually entered in lowercase.
- Flags are usually prefixed with a - (minus sign).
- More than one command can be typed on the command line if the commands are separated by a **;** (semicolon).
- Long sequences of commands can be continued on the next line by using the \ (backslash). The backslash is placed at the end of the first line. The following example shows the placement of the backslash:

  ```
  $ cat /usr/ust/mydir/mydata > \
  /usr/usts/yourdir/yourdata
  ```

When certain commands are entered, the shell prompt changes. Because some commands are actually programs (such as the **telnet** command), the prompt changes when you are operating within the command. Any command that you issue within a program is known as a subcommand. When you exit the program, the prompt returns to your shell prompt.

The operating system can operate with different shells (for example, Bourne, C, or Korn) and the commands that you enter are interpreted by the shell. Therefore, you must know what shell you are using so that you can enter the commands in the correct format.

Stopping Commands

If you enter a command and then decide to stop that command from running, you can halt the command from processing any further. To stop a command from processing, press the Interrupt key sequence (usually Ctrl-C or Alt-Pause). When the process is stopped, your shell prompt returns and you can then enter another command.

ISO 9000

ISO 9000 registered quality systems were used in the development and manufacturing of this product.

32-Bit and 64-Bit Support for the Single UNIX Specification

Beginning with Version 5.2, the operating system is designed to support The Open Group's Single UNIX Specification Version 3 (UNIX 03) for portability of UNIX-based operating systems. Many new interfaces, and some current ones, have been added or enhanced to meet this specification, making Version 5.2 even more open and portable for applications, while remaining compatible with previous releases of AIX.
To determine the proper way to develop a UNIX 03-portable application, you may need to refer to The Open Group's UNIX 03 specification, which can be accessed online or downloaded from http://www.unix.org/ .

Related Information

The following books contain information about or related to commands:

- *AIX Version 6.1 Commands Reference, Volume 1*
- *AIX Version 6.1 Commands Reference, Volume 3*
- *AIX Version 6.1 Commands Reference, Volume 4*
- *AIX Version 6.1 Commands Reference, Volume 5*
- *AIX Version 6.1 Commands Reference, Volume 6*
- *AIX Version 6.1 Files Reference*
- *Printers and printing*
- *Installation and migration*
- *AIX 5L Version 5.3 AIX Installation in a Partitioned Environment*
- *AIX Version 6.1 Network Information Services (NIS and NIS+) Guide*
- *Performance management*
- *AIX Version 6.1 Performance Tools Guide and Reference*
- *Security*
- *Networks and communication management*
- *Operating system and device management*
- *AIX Version 6.1 Technical Reference: Base Operating System and Extensions Volume 1*
- *AIX Version 6.1 Technical Reference: Base Operating System and Extensions Volume 2*
- *AIX Version 6.1 Technical Reference: Communications Volume 1*
- *AIX Version 6.1 Technical Reference: Communications Volume 2*
- *AIX Version 6.1 Technical Reference: Kernel and Subsystems Volume 1*
- *AIX Version 6.1 Technical Reference: Kernel and Subsystems Volume 2*
- *AIX Version 6.1 Web-based System Manager Administration Guide*
- *Performance Toolbox Version 2 and 3 for AIX: Guide and Reference*

Alphabetical Listing of Commands

dacinet Command

Purpose

Administers security on TCP ports in CAPP/EAL4+ configuration.

Syntax

dacinet aclflush

dacinet aclclear *Service* | *Port*

dacinet acladd *Service* | [**-**] *addr* [/*prefix_length*] [**u:***user* | *uid* | **g:***group* | *gid*]

dacinet acldel *Service* | [**-**] *addr* [/*prefix_length*] [**u:***user* | *uid* | **g:***group* | *gid*]

dacinet aclls *Service* | *Port*

dacinet setpriv *Service* | *Port*

dacinet unsetpriv *Service* | *Port*

dacinet lspriv

Description

The **dacinet** command is used to administer security on TCP ports. See the Subcommands section for details of the various functions of **dacinet**.

Subcommands

acladd Adds ACL entries to the kernel tables holding access control lists used by DACinet. The syntax of the parameters for the **acladd** subcommand is:

[**-**]*addr*[**/***length*][**u**:*user*|*uid*| **g**:*group*|*gid*]The parameters are defined as follows:

addr A DNS hostname or an IP v4/v6 address. A ″-″ before the address means that this ACL entry is used to deny access rather than to allow access.

length Indicates that *addr* is to be used as a network address rather than host address, with its first *length* bits taken from *addr*.

u:*user*|*uid*
Optional user identifier. If the *uid* is not specified, all users on the specified host or subnet are given access to the service. If supplied, only the specified user is given access.

g:*group*|*gid*
Optional group identifier. If the *gid* is not specified, all users on the specified host or subnet are given access to the service. If supplied, only the specified group is given access.

aclclear Clears the ACL for specified service or port.

© Copyright IBM Corp. 1997, 2007

acldel Deletes ACL entries from the kernel tables holding access control lists used by DACinet. The **dacinet acldel** subcommand deletes an entry from an ACL only if it is issued with parameters that exactly match the ones that were used to add the entry to the ACL. The syntax of the parameters for the **acldel** subcommands is as follows:

[**-**]*addr*[**/***length*][**u**:*user*|*uid*| **g**:*group*|*gid*]The parameters are defined as follows:

addr A DNS hostname or an IP v4/v6 address. A ″-″ before the address means that this ACL entry is used to deny access rather than to allow access.

length Indicates that *addr* is to be used as a network address rather than host address, with its first *length* bits taken from *addr*.

u:*user*|*uid*
Optional user identifier. If the *uid* is not specified, all users on the specified host or subnet are given access to the service. If supplied, only the specified user is given access.

g:*group*|*gid*
Optional group identifier. If the *gid* is not specified, all users on the specified host or subnet are given access to the service. If supplied, only the specified group is given access.

aclflush Clears all the ACLs defined in the system, rendering all TCP ports inaccessible to connection requests except from the root user on the host. It also clears privileged ports such that any process can bind to any port above 1024.

aclls Lists the ACL for the specified service or port. dacinet aclls 0 lists the default ACL. For authentication processing, from a logical perspective, the default ACL is appended to the ACL for the service. If no entry on the ACL matches the user attempting a connection to the service, access is denied. If one or more entries exist, the first one on the list with a *user*|*group*@*host*|*subnet* that matches the connection requestor determines the user's ability to connect to the service. It is thus possible to deny a service to a member of a group that has access to the service merely by adding a deny entry for that member before adding the allow entry for the group.

lspriv Lists all the privileged services or ports that are not permanently privileged (that is, it lists only privileged services with port numbers above 1024).

setpriv Makes the specified service or port privileged such that only a process with superuser privileges may bind to the port and thereby offer a service on that port. Ports below 1024 are ignored as they are permanently privileged.

unsetpriv Makes the specified service or port unprivileged such that any process may bind to it. Any process may also bind to any port in the current ephemeral port range, regardless of whether that port is marked as privileged.

Files

/usr/sbin/dacinet Contains the **dacinet** command.

---

dadmin Command

Purpose

Used to query and modify the status of the DHCP server.

Syntax

dadmin [**-?**] [**-v**] [**-h** *Hostname*] [**-n** *interval*] [**-f**] **-d** *IpAddress* | [**-x**] **-i** | [**-x**] **-s** | **-t on**|**off**|*Value* | **-q** *IpAddress* | **-r** *IpAddress* | **-p** *IpAddress* | **-c** *ClientId*

Description

The **dadmin** command lets the DHCP administrator query and modify the state of his DHCP servers' databases. It gives the administrator the ability to locally or remotely query the DHCP server for the status of an IP address, query for a pool of IP addresses, query for a client, delete an IP address mapping, refresh the server, and change the server's tracing level.

The **dadmin** command is backwards compatible with previous release DHCP servers to list their IP address status and refresh.

When querying for an IP address information, the **dadmin** command returns the IP address's status. And depending on the IP address's status, the **dadmin** command may return the lease duration, start lease time, last leased time, whether the server supports DNS A record updates for this IP address, and the client identifier which is mapped to this IP address.

When querying for a client information, the **dadmin** command returns the client's IP address and IP address status, the last time the client was given any IP address, the hostname and domain name used by the client, and whether the server supports DNS A record updates for this IP address.

When modifying the server tracing level, the **dadmin** command sets and returns the server tracing level in the form of a tracing mask. This mask represents a bitstring where each bit represents whether a specific log item is being traced by the server (see "DHCP Server Configuration File" in the online documentation). From least significant to most significant order, these log items are LOG_NONE, LOG_SYSERR, LOG_OBJERR, LOG_PROTOCOL and LOG_PROTERR (same value), LOG_WARN, AND LOG_CONFIG (same value), LOG_EVENT, and LOG_PARSEERR (same value), LOG_ACTION, LOG_INFM, LOG_ACNTING, LOG_STAT, LOG_TRACE, LOG_START, and LOG_RTRACE.

Note: LOG_START cannot be disabled. This implies a mask range from 0x0800 through 0x1FFF.

Flags

-c *ClientId*	Returns the status for a specific client that may be known to the DHCP server. *ClientId* represents the client identifier that a DHCP client used to identify itself, or the field can either be specified as hexidecimal characters only, or in the TYPE-STRING representation used by the DHCP server.
-d *IpAddress*	Deletes the lease information associated with IP address *IpAddress*. As a result, the address will be moved to the FREE state and be available for binding once again.
-f	To be used with the **-d** flag. The **-f** flag forces the deletion of the address without any prompting. Deletes the lease information associated with IP.
-h *Hostname*	Used to specify the destination DHCP server. *Hostname* can either be a name or IP address.
-i	Reinitializes the DHCP server. This flag signals the server to sync its databases and restarts by rereading the configuration file.
-n *interval*	Displays server statistics, summaries, and any requested intervals.
-p *IpAddress*	Returns the status of each address in a subnet. *IpAddress* is used to identify the subnet to a list.
-q *IpAddress*	Returns the status of a specific IP address.
-r *IpAddress*	Puts the IP address in the `Free` state.
-s	Returns the status of each address in the DHCP server's configured pools.
-t on|off| *Value*	Changes the tracing level of the DHCP server. Trace values are reported in a hexidecimal format representing the tracing mask in use on the server. *Value* can be specified as either a decimal or hexidecimal format. The keywords **on** and **off** enable or disable a single bit at a time in the tracing mask.
-v	Executes the command in verbose mode.
-x	Use Version 1 of the **dadmin** protocol. The **-x** flag is used to connect to previous release DHCP servers and is only valid for the **-i** and **-s** flags. Follow with `6` when connecting with DHCPv6 server.
-?	Displays the usage syntax.

Exit Status

0	Successful completion.
>0	An error occurred.

Security

To secure connections from the dadmin clients, the DHCP server only allows connections from the server itself or from remote machines that are included in the superuser's **.rhosts** file. To prevent ordinary users from modifying the DHCP server's address mappings, the administrator should ensure that the execution of the **dadmin** command is limited to the proper users on those machines that are allowed access.

Files

/usr/sbin/dadmin	Contains the dadmin command.

Related Information

The **.rhosts** file format, **DHCP Server Configuration File** in the *AIX Version 6.1 Files Reference*.

The **dhcpsd** daemon.

TCP/IP address and parameter assignment - Dynamic Host Configuration Protocol in the *Networks and communication management*.

TCP/IP daemons in the *Networks and communication management*

---

date Command

Purpose

Displays or sets the date or time.

Syntax

To Set the Date and Time as Root User

/usr/bin/date [**-n**] [**-u**] [*Date*] [**+***FieldDescriptor ...*]

To Display the Date and Time

/usr/bin/date [**-u**] [**+***FieldDescriptor* ...]

To adjust the Time in Seconds as root User

/usr/bin/date [**-a** [**+** | **-**]*sss*[*.fff*]

Description

Attention: Do not change the date when the system is running with more than one user.

The **date** command writes the current date and time to standard output if called with no flags or with a flag list that begins with a **+** (plus sign). Otherwise, it sets the current date. Only a root user can change the date and time. The **date** command prints out the usage message on any unrecognized flags or input.

The following formats can be used when setting the date with the *Date* parameter:

- *mmddHHMM*[*YYyy*]
- *mmddHHMM*[*yy*]

The variables to the *Date* parameter are defined as follows:

mm Specifies the month number.

dd Specifies the number of the day in the month.

HH Specifies the hour in the day (using a 24-hour clock).

MM Specifies the minute number.

YY Specifies the first two digits of the year.
Note: If you do not specify the first two digits of the year, values in the range 69 to 99 refer to the twentieth century, 1969 to 1999 inclusive, and values in the range 00 to 68 refer to years in the twenty-first century, 2000 to 2068 inclusive.

yy Specifies the last two digits of the year.
Note: The **date** command accepts a 4 digit year as input. For example, if a four-digit year is specified, the **date** command tries to set the year to ″YYyy″ and fails for values which are out of range (less than 1970 and greater than 2037).

The current year is used as the default value when the year is not specified. The system operates in Coordinated Universal Time (CUT).

If you follow the **date** command with a **+** (plus sign) and a field descriptor, you can control the output of the command. You must precede each field descriptor with a **%** (percent sign). The system replaces the field descriptor with the specified value. Enter a literal % as %% (two percent signs). The **date** command copies any other characters to the output without change. The **date** command always ends the string with a new-line character.

Flags

-a [**+** | **-**]*sss*[*.fff*] Slowly adjusts the time by ***sss.fff*** seconds (*fff* represents fractions of a second). This adjustment can be positive or negative. The system's clock will be sped up or slowed down until it has drifted by the number of seconds specified.

-n Does not set the time globally on all machines in a local area network that have their clocks synchronized.

-u Displays or sets the time in Coordinated Universal Time (CUT).

Field Descriptors

%a Displays the locale's abbreviated weekday name.

%A Displays the locale's full weekday name.

%b Displays the locale's abbreviated month name.

%B Displays the locale's full month name.

%c Displays the locale's appropriate date and time representation. This is the default.

%C Displays the first two digits of the four-digit year as a decimal number (00-99). A year is divided by 100 and truncated to an integer.

%d Displays the day of the month as a decimal number (01-31). In a two-digit field, a 0 is used as leading space fill.

%D Displays the date in the format equivalent to **%m/%d/%y**.

%e Displays the day of the month as a decimal number (1-31). In a two-digit field, a blank space is used as leading space fill.

%h Displays the locale's abbreviated month name (a synonym for **%b**).

%H Displays the hour (24-hour clock) as a decimal number (00-23).

%I Displays the hour (12-hour clock) as a decimal number (01-12).

%j Displays the day of year as a decimal number (001-366).

%k Displays the 24-hour-clock hour clock as a right-justified, space-filled number (0 to 23).

%m Displays the month of year as a decimal number (01-12).

%M Displays the minutes as a decimal number (00-59).

%n Inserts a <new-line> character.

%p Displays the locale's equivalent of either AM or PM.

%r Displays 12-hour clock time (01-12) using the AM-PM notation; in the POSIX locale, this is equivalent to **%I:%M:%S %p**.
%S Displays the seconds as a decimal number (00- 59).
%s Displays the number of seconds since January 1, 1970, Coordinated Universal Time (CUT).
%t Inserts a <tab> character.
%T Displays the 24-hour clock (00-23) in the format equivalent to **HH:MM:SS** .
%u Displays the weekday as a decimal number from 1-7 (Sunday = 7). Refer to the **%w** field descriptor.
%U Displays week of the year(Sunday as the first day of the week) as a decimal number[00 - 53] . All days in a new year preceding the first Sunday are considered to be in week 0.
%V Displays the week of the year as a decimal number from 01-53 (Monday is used as the first day of the week). If the week containing January 1 has four or more days in the new year, then it is considered week 01; otherwise, it is week 53 of the previous year.
%w Displays the weekday as a decimal number from 0-6 (Sunday = 0). Refer to the **%u** field descriptor.
%W Displays the week number of the year as a decimal number (00-53) counting Monday as the first day of the week.
%x Displays the locale's appropriate date representation.
%X Displays the locale's appropriate time representation.
%y Displays the last two numbers of the year (00-99).
%Y Displays the four-digit year as a decimal number.
%Z Displays the time-zone name, or no characters if no time zone is determinable.
%% Displays a % (percent sign) character.

Modified Field Descriptors

The **%E** and **%O** field descriptors can be modified to indicate a different format or specification, as described in **LC_TIME** Categoryi for the Locale Definition Source File Format in *AIX Version 6.1 Files Reference*. If the corresponding keyword (see the **era**, **era_year**, **era_d_fmt**, and **alt_digits** keywords) is not specified or not supported for the current locale, the unmodified field descriptor value is used.

%Ec Displays the locale's alternative appropriate date and time representation.
%EC Displays the name of the base year (or other time period) in the locale's alternative representation.
%Ex Displays the locale's alternative date representation.
%EX Displays the locale's alternative time representation.
%Ey Displays the offset from the **%EC** field descriptor (year only) in the locale's alternative representation.
%EY Displays the full alternative year representation.
%Od Displays the day of the month using the locale's alternative numeric symbols.
%Oe Displays the day of the month using the locale's alternative numeric symbols.
%OH Displays the hour (24-hour clock) using the locale's alternative numeric symbols.
%OI Displays the hour (12-hour clock) using the locale's alternative numeric symbols.

%Om Displays the month using the locale's alternative numeric symbols.
%OM Displays minutes using the locale's alternative numeric symbols.
%OS Displays seconds using the locale's alternative numeric symbols.
%Ou Displays the weekday as a number in the locale's alternative representation (Monday=1).
%OU Displays the week number of the year using the locale's alternative numeric symbols. Sunday is considered the first day of the week.
%OV Displays the week number of the year using the locale's alternative numeric symbols. Monday is considered the first day of the week.
%Ow Displays the weekday as a number in the locale's alternative representation (Sunday =0).
%OW Displays the week number of the year using the locale's alternative numeric symbols. Monday is considered the first day of the week.
%Oy Displays the year (offset from %C) in alternative representation.

Exit Status

This command returns the following exit values:

0 The date was written successfully.
>0 An error occurred.

Security

Attention RBAC users and Trusted AIX users: This command can perform privileged operations. Only privileged users can run privileged operations. For more information about authorizations and privileges, see Privileged Command Database in *Security*. For a list of privileges and the authorizations associated with this command, see the **/etc/security/privcmds** database file.

Examples

1. To display current date and time, enter:

   ```
   date
   ```

2. To set the date and time, enter:

   ```
   date 0217142590
   ```

   For a system using CST as its time zone, this sets the date and time to Sat Feb 17 14:25:00 CST 1990.

   Note: You must have root authority to change the date and time.

3. To display the date and time in a specified format, enter:

   ```
   date +"%r %a %d %h %y (Julian Date: %j)"
   ```

   This displays the date shown in Example 2 as:

   ```
   02:25:03 PM Fri 17 Feb 90 (Julian Date: 048)
   ```

Environment Variables

The following environment variables affect the execution of the **date** command.

LANG	Determines the locale to use when both **LC_ALL** and the corresponding environment variable (beginning with **LC**_) do not specify a locale.
LC_ALL	Determines the locale to be used to override any values for locale categories specified by the setting of **LANG** or any environment variable beginning with **LC**_.
LC_CTYPE	Determines the locale for the interpretation of sequences of bytes of text data as characters (for example, single versus multibyte character in an argument).
LC_MESSAGES	Determines the language in which messages should be written.
LC_TIME	Determines the contents of date and time strings written by **date**.
NLSPATH	Determines the location of message catalogues for the processing of **LC_MESSAGES**.
TZ	Specifies the time zone in which the time and date are written, unless the **-u** option is specified. If the **TZ** variable is not set and the **-u** flag is not specified, an unspecified system default time zone is used.

Related Information

The **environment** file.

The **localtime** subroutine, **strftime** subroutine, **time** subroutine.

LC_TIME Category for the Locale Definition Source File Format in *AIX Version 6.1 Files Reference*.

Understanding Locale in *AIX Version 6.1 National Language Support Guide and Reference* discusses locale values.

dbts Command

Purpose
Debugs a thin server.

Syntax
dbts [**-v**] *ThinServer*

Description
The **dbts** command lets a thin server boot into debug mode. The command checks if the thin server was previously booted into debug mode by searching for a debug boot image created for the thin server. If none is found, the common image that the thin server is using is cloned and a debug boot image is created from the clone to allow the thin server to boot into debug mode. The debug boot image clone uses the following naming convention:

{*COSI name*}_{*thin server name*}-debug

After the thin server is finished using the debug common image, the **swts** command must be run to switch the thin server to a different common image. The **rmcosi** command removes the debug common image created from the **dbts** command. The **dbts** command can run on either a NIM master or a thin server.

Flags

-v	Enables verbose debug output while the **dbts** command runs.

Exit Status

0	The command completed successfully.
>0	An error occurred.

Security
Access Control: You must have root authority to run the **dbts** command.

Examples
1. To debug boot a thin server named `lobo` that is using a common image named `cosi1`, enter:

   ```
   dbts lobo
   ```

   A debug boot image named `cosi1_lobo-debug` is created to boot `lobo` into debug mode.

Location
/usr/sbin/dbts

Files

/etc/niminfo	Contains variables used by NIM.

Related Information

The **lsts** command, **mkcosi** command, **mkts** command, **nim** command, **nim_clients_setup** command, **nim_master_setup** command, **nimconfig** command, **rmts** command, **swts** command.

---

dbx Command

Purpose

Provides an environment to debug and run programs.

Syntax

dbx [**-a** *ProcessID*] [**-c** *CommandFile*] [**-d** *NestingDepth*] [**-I** *Directory*] [**-E** *DebugEnvironment*] [**-p** *oldpath=newpath:...| pathfile*] [**-u**] [**-F**] [**-r**] [**-x**] [*ObjectFile* [*CoreFile*]]

Description

The **dbx** command provides a symbolic debug program for C, C++, Pascal, and FORTRAN programs, allowing you to carry out operations such as the following:

- Examine object and core files.
- Provide a controlled environment for running a program.
- Set breakpoints at selected statements or run the program one line at a time.
- Debug using symbolic variables and display them in their correct format.

The *ObjectFile* parameter is an object (executable) file produced by a compiler. Use the **-g** (generate symbol table) flag when compiling your program to produce the information the **dbx** command needs.

> **Note:** The **-g** flag of the **cc** command should be used when the object file is compiled. If the **-g** flag is not used or if symbol references are removed from the **xcoff** file with the **strip** command, the symbolic capabilities of the **dbx** command are limited. In addition, do not use the **-O** compiler option to optimize an executable that you plan to debug with **dbx**. Optimization rearranges the code and compromises the debug data, further limiting the value of debugging the executable with **dbx**.

If the **-c** flag is not specified, the **dbx** command checks for a **.dbxinit** file in the user's **$HOME** directory. It then checks for a **.dbxinit** file in the user's current directory. If a **.dbxinit** file exists in the current directory, that file overrides the **.dbxinit** file in the user's **$HOME** directory. If a **.dbxinit** file exists in the user's **$HOME** directory or current directory, that file's subcommands run at the beginning of the debug session. Use an editor to create a **.dbxinit** file.

If *ObjectFile* is not specified, then **dbx** asks for the name of the object file to be examined. The default is **a.out**. If the **core** file exists in the current directory or a *CoreFile* parameter is specified, then **dbx** reports the location where the program faulted. Variables, registers, and memory held in the core image may be examined until execution of *ObjectFile* begins. At that point the **dbx** debug program prompts for commands.

Expression Handling

The **dbx** program can display a wide range of expressions. You can specify expressions in the **dbx** debug program with a common subset of C and Pascal syntax, with some FORTRAN extensions.

The following operators are valid in the debug program:

***** (asterisk) or **^** (caret)	Denotes indirection or pointer dereferencing.
[ ] (brackets) or **( )** (parentheses)	Denotes subscript array expressions.

. (period)	Use this field reference operator with pointers and structures. This makes the C operator -> (arrow) unnecessary, although it is allowed.
& (ampersand)	Gets the address of a variable.
.. (two periods)	Separates the upper and lower bounds when specifying a subsection of an array. For example: **n[1..4]**.

The following types of operations are valid in expressions in the debug program:

Algebraic	**=**, **-**, *****, **/** (floating division), **div** (integral division), **mod**, **exp** (exponentiation)
Bitwise	**-**, **|**, **bitand**, **xor**, **~**. **<<**, **>>**
Logical	**or**, **and**, **not**, **||**, **&&**
Comparison	**<**, **>**, **<=**, **>=**, **< >** or **!=**, **=** or **==**
Other	**(typename),sizeof**

Logical and comparison expressions are allowed as conditions in **stop** and **trace**.

Expression types are checked. You override an expression type by using a renaming or casting operator. The three forms of type renaming are *Typename(Expression)*, *Expression*|*Typename*, and *(Typename) Expression*. The following is an example where the *x* variable is an integer with value 97:

```
(dbx) print x
97
(dbx) print char (x), x \ char, (char) x, x
'a' 'a' 'a' 97
```

Command Line Editing

The **dbx** commands provides a command line editing feature similar to those provide by Korn Shell. **vi** mode provides **vi-like** editing features, while **emacs** mode gives you controls similar to **emacs**.

These features can be turned on by using **dbx** subcommand **set -o** or **set edit**. To turn on vi-style command-line editing, you would type the subcommand **set edit vi** or **set -o vi**.

You can also use the **EDITOR** environment variable to set the editing mode.

The **dbx** command saves commands entered to a history file **.dbxhistory**. If the **DBXHISTFILE** environment variable is not set, the history file used is **$HOME/.dbxhistory**.

By default, **dbx** saves the text of the last 128 commands entered. The **DBXHISTSIZE** environment variable can be used to increase this limit.

Flags

-a *ProcessID*	Attaches the debug program to a process that is running. To attach the debug program, you need authority to send signals to this process. Use the **ps** command to determine the process ID. If you have permission, the **dbx** program interrupts the process using the **ptrace** system call to send a **SIGTRAP** signal to the process, which cannot ignore the **SIGTRAP** signal. It then determines the full name of the object file, reads in the symbolic information, and prompts for commands.

-c *CommandFile* — Runs the **dbx** subcommands in the file before reading from standard input. The specified file in the **$HOME** directory is processed first; then the file in the current directory is processed. The command file in the current directory overrides the command file in the **$HOME** directory. If the specified file does not exist in either the **$HOME** directory or the current directory, a warning message is displayed. The **source** subcommand can be used once the **dbx** program is started.

-d *NestingDepth* — Sets the limit for the nesting of program blocks. The default nesting depth limit is 25.

-E *DebugEnvironment* — Specifies the environment variable for the debug program.

-p *oldpath=newpath:...| pathfile* — Specifies a substitution for library paths when examining core files in the format *oldpath=newpath*. *oldpath* specifies the value to be substituted (as stored in the core file) and *newpath* specifies what it is to be replaced with. These may be complete or partial, relative or absolute paths. Multiple substitutions may be specified, separated by colons. Alternatively, the **-p** flag may specify the name of a file from which mappings in the previously described format are to be read. Only one mapping per line is allowed when mappings are read from a file.

-F — Can be used to turn off the lazy read mode and make the **dbx** command read all symbols at startup time. By default, lazy reading mode is on: it reads only required symbol table information on initiation of **dbx** session. In this mode, **dbx** will not read local variables and types whose symbolic information has not been read. Therefore, commands such as **whereis i** may not list all instances of the local variable **i** in every function.

-I *Directory* — (Uppercase i) Includes directory specified by the *Directory* variable in the list of directories searched for source files. The default is to look for source files in the following directories:

- The directory the source file was located in when it was compiled. This directory is searched only if the compiler placed the source path in the object.
- The current directory.
- The directory where the program is currently located.

-r — Runs the object file immediately. If it terminates successfully, the **dbx** debug program is exited. Otherwise, the debug program is entered and the reason for termination is reported.

> **Note:** Unless **-r** is specified, the **dbx** command prompts the user and waits for a command.

-u — Causes the **dbx** command to prefix file name symbols with an @ (at sign). This flag reduces the possibility of ambiguous symbol names.

-x — Prevents the **dbx** command from stripping _ (trailing underscore) characters from symbols originating in FORTRAN source code. This flag allows **dbx** to distinguish between symbols which are identical except for an underscore character, such as xxx and xxx_.

Examples

1. The following example explains how to start the **dbx** debug program simultaneously with a process. The example uses a program called **samp.c**. This C program is first compiled with the **-g** flag to produce an object file that includes symbolic table references. In this case, the program is named **samp**:

   ```
   $ cc -g samp.c -o samp
   ```

   When the program **samp** is run, the operating system reports a bus error and writes a core image to your current working directory as follows:

   ```
   $ samp
   Bus Error - core dumped
   ```

To determine the location where the error occurred, enter:

```
$ dbx samp
```

The system returns the following message:

```
dbx version 3.1
Type 'help' for help.
reading symbolic information . . . [
using memory image in core]
  25   x[i] = 0;
(dbx) quit
```

2. This example explains how to attach **dbx** to a process. This example uses the following program, **looper.c**:

```
main()
{
      int i,x[10];

      for (i = 0; i < 10;);
}
```

The program will never terminate because **i** is never incremented. Compile **looper.c** with the **-g** flag to get symbolic debugging capability:

```
$ cc -g looper.c -o looper
```

Run **looper** from the command line and perform the following steps to attach **dbx** to the program while it is running:

a. To attach **dbx** to **looper**, you must determine the process ID. If you did not run **looper** as a background process, you must have another Xwindow open. From this Xwindow , enter:

```
ps -u UserID
```

where *UserID* is your login ID. All active processes that belong to you are displayed as follows:

```
PID     TTY      TIME    COMMAND
68      console   0:04    sh
467     lft3     10:48    looper
```

In this example the process ID associated with **looper** is 467.

b. To attach **dbx** to **looper**, enter:

```
$ dbx -a 467
```

The system returns the following message:

```
Waiting to attach to process 467 . . .
Successfully attached to /tmp/looper.
dbx is initializing
Type 'help' for help.
reading symbolic information . . .

attached in main at line 5
5     for (i = 0; i < 10;);
(dbx)
```

You can now query and debug the process as if it had been originally started with **dbx**.

3. To add directories to the list of directories to be searched for the source file of an executable file **objfile**, you can enter:

```
$dbx -I /home/user/src -I /home/group/src
objfile
```

The **use** subcommand may be used for this function once **dbx** is started. The **use** command resets the list of directories, whereas the **-I** flag adds a directory to the list.

4. To use the **-r** flag, enter:

```
$ dbx -r samp
```

The system returns the following message:

```
Entering debug program . . .
dbx version 3.1
Type 'help' for help.
reading symbolic information . . .
bus error in main at line 25
  25   x[i] = 0;
(dbx) quit
```

The **-r** flag allows you to examine the state of your process in memory even though a core image is not taken.

5. To specify the environment variables for the debug program, enter:

```
dbx -E LIBPATH=/home/user/lib -E LANG=Ja_JP objfile
```

dbx Subcommands

Note: The subcommands can only be used while running the **dbx** debug program.

/	Searches forward in the current source file for a pattern.
?	Searches backward in the current source file for a pattern.
addcmd	Adds **dbx** subcommands to the specified event numbers.
alias	Creates aliases for **dbx** subcommands.
assign	Assigns a value to a variable.
attribute	Displays information about all or selected attributes objects.
call	Runs the object code associated with the named procedure or function.
case	Changes how the dbx debug program interprets symbols.
catch	Starts trapping a signal before that signal is sent to the application program.
clear	Removes all stops at a given source line.
cleari	Removes all breakpoints at an address.
condition	Displays information about all or selected condition variables.
cont	Continues application program execution from the current stopping point until the program finishes or another breakpoint is encountered.
corefile	Displays high-level data about a corefile.
delcmd	Deletes **dbx** subcommands associated with the specified event number.
delete	Removes the traces and stops corresponding to the specified event numbers and **tskip** counts for a thread.
detach	Continues execution of application and exits the debug program.
disable	Disables the traces and stops corresponding to the specified event numbers.
display memory	Displays the contents of memory.
down	Moves the current function down the stack.
dump	Displays the names and values of variables in the specified procedure.
edit	Starts an editor on the specified file.
enable	Enables the traces and stops corresponding to the specified event numbers.
fd	Displays file descriptor information.
file	Changes the current source file to the specified file.
frame	Changes the current function to the function corresponding to the specified stack frame number.
func	Changes the current function to the specified procedure or function.
goto	Causes the specified source line to be the next line run.
gotoi	Changes the program counter address.
handler	Displays information about pthreads atfork or cancellation cleanup handlers.
help	Displays help information for **dbx** subcommands or topics.
ignore	Stops trapping a signal before that signal is sent to the application program.
kthread	Displays information about kernel threads.
list	Displays lines of the current source file.
listi	Lists instructions from the application program.

malloc	Displays information about the program's usage of the malloc subsystem.
map	Displays information about load characteristics of the application.
move	Changes the next line to be displayed.
multproc	Enables or disables multiprocess debugging.
mutex	Displays information about all or selected mutexes.
next	Runs the application program up to the next source line.
nexti	Runs the application program up to the next machine instruction.
onceblock	Displays information about once blocks.
plugin	Invokes a plug-in subcommand or displays the names of available plug-ins.
pluginload	Loads a plug-in.
pluginunload	Unloads a plug-in.
print	Prints the value of an expression or runs a procedure and prints the return code of that procedure.
proc	Displays information about the process.
prompt	Changes the dbx command prompt.
quit	Stops the dbx debug program.
registers	Displays the values of all general-purpose registers, system-control registers, floating-point registers, and the current instruction register.
rerun	Begins execution of an application with the previous arguments.
resource	Displays information about resources owned or waited on by pthreads.
return	Continues running the application program until a return to the specified procedure is reached.
rwlock	Displays information about the rwlocks.
run	Begins running an application.
screen	Opens an Xwindow for dbx command interaction.
set	Defines a value for a dbx debug program variable.
sh	Passes a command to the shell to be run.
skip	Continues running the application program from the current stopping point.
source	Reads **dbx** subcommands from a file.
status	Displays the active trace, stop subcommands, and remaining thread **tskip** counts.
step	Runs one source line.
stepi	Runs one machine instruction.
stophwp	Sets a hardware watchpoint stop.
stop	Stops running of the application program.
stopi	Sets a stop at a specified location.
thread	Displays and controls threads.
tls	Displays TLS initialization template information.
tnext	Runs a thread up to the next source line.
tnexti	Runs a thread up to the next machine instruction.
trace	Prints tracing information.
tracehwp	Sets a hardware watchpoint trace.
tracei	Turns on tracing.
tskip	Skips breakpoints for a thread.
tstep	Runs a thread for one source line.
tstepi	Runs a thread for one machine instruction.
tstop	Sets a source-level breakpoint stop for a thread.
tstophwp	Sets a thread-level hardware watchpoint stop.
tstopi	Sets an instruction-level breakpoint stop for a thread.
ttrace	Sets a source-level trace for a thread.
ttracehwp	Sets a thread-level hardware watchpoint trace.
ttracei	Sets an instruction-level trace for a thread.
unalias	Removes an alias.
unset	Deletes a variable.
up	Moves the current function up the stack.
use	Sets the list of directories to be searched when looking for source files.

whatis	Displays the declaration of application program components.
where	Displays a list of active procedures and functions.
whereis	Displays the full qualifications of all the symbols whose names match the specified identifier.
which	Displays the full qualification of the given identifier.

/ Subcommand

/ [*RegularExpression* [**/**]]

The **/** subcommand searches forward in the current source file for the pattern specified by the *RegularExpression* parameter. Entering the **/** subcommand with no arguments causes **dbx** to search forward for the previous regular expression. The search wraps around the end of the file.

Examples:

1. To search forward in the current source file for the number 12, enter:
   ```
   / 12
   ```
2. To repeat the previous search, enter:
   ```
   /
   ```

See the **?** (search) subcommand and the **regcmp** subroutine.

? Subcommand

? [*RegularExpression* [**?**]]

The **?** subcommand searches backward in the current source file for the pattern specified by the *RegularExpression* parameter. Entering the **?** subcommand with no arguments causes the **dbx** command to search backwards for the previous regular expression. The search wraps around the end of the file.

Examples:

1. To search backward in the current source file for the letter `z`, enter:
   ```
   ?z
   ```
2. To repeat the previous search, enter:
   ```
   ?
   ```

See the **/** (search) subcommand and the **regcmp** subroutine.

addcmd Subcommand

addcmd { *Number*... | **all** } "*commands_string*"

The **addcmd** subcommand adds **dbx** subcommands to the specified event, which will be executed whenever the breakpoint, tracepoint, or watchpoint corresponding to the event is hit. The **dbx** subcommands can be specified through the *"commands_string"* parameter, which is a group of **dbx** subcommands separated by a semicolon (;). The event to which the **dbx** subcommands are to be added can be specified through the *Number* parameter, or the **dbx** subcommands can be added to all events by using the **all** flag.

Flags:

all	Adds **dbx** subcommands to all the events.

Examples:

1. To add the **where** subcommand to event number 1, enter:
   ```
   addcmd 1 "where"
   ```
2. To add the **registers** subcommand to event number 2, enter:
   ```
   addcmd 2 "registers"
   ```
3. To add the **where** and **registers** subcommands to event number 3, enter:
   ```
   addcmd 3 "where;registers"
   ```

See **clear** subcommand, the **delcmd** subcommand, the **delete** subcommand, **disable** subcommand, **enable** subcommand, the **stop** subcommand, the **status** subcommand, and the **trace** subcommand. Also see Setting and Deleting Breakpoints in in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

alias Subcommand

alias [*Name* [[(*Arglist*)] *String* | *Subcommand*]]

The **alias** subcommand creates aliases for **dbx** subcommands. The *Name* parameter is the alias being created. The *String* parameter is a series of **dbx** subcommands that, after the execution of this subcommand, can be referred to by *Name*. If the **alias** subcommand is used without parameters, it displays all current aliases.

Examples:

1. To substitute `rr` for `rerun`, enter:
   ```
   alias rr rerun
   ```
2. To run the two subcommands `print n` and `step` whenever `printandstep` is typed at the command line, enter:
   ```
   alias printandstep "print n; step"
   ```
3. The `alias` subcommand can also be used as a limited macro facility. For example:
   ```
   (dbx) alias px(n) "set $hexints; print n; unset $hexints"
   (dbx) alias a(x,y) "print symname[x]->symvalue._n_n.name.Id[y]"
   (dbx) px(126)
   0x7e
   ```
   In this example, the alias `px` prints a value in hexadecimal without permanently affecting the debugging environment.

assign Subcommand

assign *Variable*=*Expression*

The **assign** subcommand assigns the value specified by the *Expression* parameter to the variable specified by the *Variable* parameter.

Examples:

1. To assign a value of `5` to the `x` variable, enter:
   ```
   assign x = 5
   ```
2. To assign the value of the `y` variable to the `x` variable, enter:
   ```
   assign x =  y
   ```
3. To assign the character value `'z'` to the `z` variable, enter:
   ```
   assign  z  =  'z'
   ```
4. To assign the boolean value `false` to the logical type variable `B`, enter:
   ```
   assign  B  =  false
   ```
5. To assign the `"Hello World"` string to a character pointer `Y`, enter:
   ```
   assign  Y  =  "Hello  World"
   ```

6. To disable type checking, set the **dbx** debug program variable `$unsafeassign` by entering:
   ```
   set $unsafeassign
   ```

See Displaying and Modifying Variables.

attribute Subcommand

attribute [*AttributeNumber* ...]

The **attribute** subcommand displays information about the user thread, mutex, or condition attributes objects defined by the *AttributeNumber* parameters. If no parameters are specified, all attributes objects are listed.

For each attributes object listed, the following information is displayed:

`attr`	Indicates the symbolic name of the attributes object, in the form `$a`*AttributeNumber*.
`obj_addr`	Indicates the address of the attributes object.
`type`	Indicates the type of the attributes object; this can be `thr`, `mutex`, or `cond` for user threads, mutexes, and condition variables respectively.
`state`	Indicates the state of the attributes object. This can be `valid` or `inval`.
`stack`	Indicates the stacksize attribute of a thread attributes object.
`scope`	Indicates the scope attribute of a thread attributes object. This determines the contention scope of the thread, and defines the set of threads with which it must contend for processing resources. The value can be `sys` or `pro` for system or process contention scope.
`prio`	Indicates the priority attribute of a thread attributes object.
`sched`	Indicates the schedpolicy attribute of a thread attributes object. This attribute controls scheduling policy, and can be `fifo` , `rr` (round robin), or `other`.
`p-shar`	Indicates the process-shared attribute of a mutex or condition attribute object. A mutex or condition is process-shared if it can be accessed by threads belonging to different processes. The value can be `yes` or `no`.
`protocol`	Indicates the protocol attribute of a mutex. This attribute determines the effect of holding the mutex on a threads priority. The value can be `no_prio`, `prio`, or `protect`.
`clock`	Indicates the clock attribute of a condition attribute object. This attribute determines which clock must be used when a thread that waits for the condition variable as specified a timeout. The value can be `realtime` or `monotonic`.

Notes:

1. The **print** subcommand of the **dbx** debug program recognizes symbolic attribute names, and can be used to display the status of the corresponding object.
2. The available attributes depend on the implementation of POSIX options.

Examples:

1. To list information about all attributes, enter:
   ```
   attribute
   ```

   The output is similar to:
   ```
   attr   obj_addr   type  state  stack   scope    prio
   sched p-shar
   $a1   0x200035c8  mutex valid                                 no
   $a2   0x20003628  cond  valid                                 no
   $a3   0x200037c8  thr   valid  57344    sys      126 other
   $a4   0x200050f8  thr   valid  57344    pro      126 other
   ```
2. To list information about attributes 1 and 3, enter:
   ```
   attribute 1 3
   ```

   The output is similar to:

```
attr   obj_addr   type  state  stack   scope    prio
sched p-shar
$a1   0x200035c8  mutex valid                              no
$a3   0x200037c8  thr   valid  57344    sys      126 other
```

See the **condition** subcommand, **mutex** subcommand, **print** subcommand, and **thread** subcommand for the **dbx** command.

Also, see Creating Threads, Using Mutexes, and Using Condition Variables in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

call Subcommand

call *Procedure* ([*Parameters*])

The **call** subcommand runs the procedure specified by the *Procedure* parameter. The return code is not printed. If any parameters are specified, they are passed to the procedure being run.

Note: The **call** subcommand cannot be used to call functions that take vector parameters.

Example: To call a command while running `dbx`, enter:

```
(dbx) call printf("hello")
hello
```

printf returns successfully.

case Subcommand

case [**default** | **mixed** | **lower** | **upper**]

The **case** subcommand changes how the **dbx** debug program interprets symbols. The default handling of symbols is based on the current language. If the current language is C, C++, or undefined, the symbols are not folded; if the current language is FORTRAN or Pascal, the symbols are folded to lowercase. Use this subcommand if a symbol needs to be interpreted in a way not consistent with the current language.

Entering the **case** subcommand with no parameters displays the current case mode.

Flags:

default	Varies with the current language.
mixed	Causes symbols to be interpreted as they actually appear.
lower	Causes symbols to be interpreted as lowercase.
upper	Causes symbols to be interpreted as uppercase.

Examples:

1. To display the current case mode, enter:
   ```
   case
   ```
2. To instruct **dbx** to interpret symbols as they actually appear, enter:
   ```
   case mixed
   ```
3. To instruct **dbx** to interpret symbols as uppercase, enter:
   ```
   case upper
   ```

See Folding Variables to Lowercase and Uppercase.

catch Subcommand

catch [*SignalNumber* | *SignalName*]

The **catch** subcommand starts the trapping of a specified signal before that signal is sent to the application program. This subcommand is useful when the application program being debugged handles signals such as interrupts. The signal to be trapped can be specified by number or by name using either the *SignalNumber* or the *SignalName* parameter, respectively. Signal names are case insensitive, and the **SIG** prefix is optional. If neither the *SignalNumber* nor the *SignalName* parameter is specified, all signals are trapped by default except the **SIGHUP**, **SIGCLD**, **SIGALARM**, and **SIGKILL** signals. If no arguments are specified, the current list of signals to be caught is displayed.

Examples:

1. To display a current list of signals to be caught by **dbx**, enter:
   ```
   catch
   ```
2. To trap signal `SIGALARM`, enter:
   ```
   catch SIGALARM
   ```

See the **ignore** subcommand and Handling Signals.

clear Subcommand

clear *SourceLine*

The **clear** subcommand removes all stops at a given source line. The *SourceLine* parameter can be specified in two formats:

- As an integer
- As a file name string followed by a : (colon) and an integer

Examples: To remove breakpoints set at line `19`, enter:

```
clear 19
```

The **cleari** subcommand and **delete** subcommand. Also, see Setting and Deleting Breakpoints in in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

cleari Subcommand

cleari *Address*

The **cleari** subcommand clears all the breakpoints at the address specified by the *Address* parameter.

Examples:

1. To remove a breakpoint set at address `0x100001b4`, enter:
   ```
   cleari 0x100001b4
   ```
2. To remove a breakpoint set at the `main()` procedure address, enter:
   ```
   cleari &main
   ```

See the **clear** subcommand, the **delete** subcommand, and Setting and Deleting Breakpoints in in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

condition Subcommand

condition [**wait** | **nowait** | *ConditionNumber* ...]

The **condition** subcommand displays information about one or more condition variables. If one or more *ConditionNumber* parameters are given, the **condition** subcommand displays information about the specified condition variables. If no flags or parameters are specified, the **condition** subcommand lists all condition variables.

The information listed for each condition is as follows:

`cv`	Indicates the symbolic name of the condition variable, in the form `$c`*ConditionNumber*.
`obj_addr`	Indicates the memory address of the condition variable.
`clock`	Indicates the clock attribute of the condition variable.
`num_wait`	Indicates the number of threads waiting on the condition variable.
`waiters`	Lists the user threads which are waiting on the condition variable.

> **Note:** The **print** subcommand of the **dbx** debug program recognizes symbolic condition variable names, and can be used to display the status of the corresponding object.

Flags:

wait	Displays condition variables which have waiting threads.
nowait	Displays condition variables which have no waiting threads.

Examples:

1. To display information about all condition variables, enter:
   ```
   condition
   ```
2. To display information about all condition variables which have waiting threads, enter:
   ```
   condition  wait
   ```
3. To display information about the condition variable 3, enter:
   ```
   condition 3
   ```
   The output is similar to:
   ```
   cv      obj_addr     num_wait  waiters
   $c3     0x20003290         0
   ```

See the **attribute** subcommand, **mutex** subcommand, **print** subcommand, and **thread** subcommand.

Also, see Using Condition Variables in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*:.

cont Subcommand

cont [*SignalNumber* | *SignalName*]

The **cont** subcommand continues the execution of the application program from the current stopping point until either the program finishes or another breakpoint is reached. If a signal is specified, either by the number specified in the *SignalNumber* parameter or by the name specified in the *SignalName* parameter, the program continues as if that signal had been received. Signal names are not case sensitive and the **SIG** prefix is optional. If no signal is specified, the program continues as if it had not been stopped.

Examples

1. To continue program execution from current stopping point, enter:
   ```
   cont
   ```
2. To continue program execution as though it received the signal SIGQUIT, enter:
   ```
   cont SIGQUIT
   ```

See the **detach** subcommand for the **dbx** command, the **goto** subcommand for the **dbx** command, the **next** subcommand for the **dbx** command, the **skip** subcommand for the **dbx** command, the **step** subcommand for the **dbx** command.

corefile Subcommand

The **corefile** subcommand displays information from the header of a corefile, including the executable name, corefile format versioning information, flags indicating which data is available, the signal that caused the crash, and the execution mode of the process that dumped core.

delcmd Subcommand

delcmd *EventNumber* { *Number...* | **all** }

The **delcmd** subcommand removes the **dbx** subcommands associated with the specified event. The **dbx** subcommands to be removed can be specified through *Number* parameters, or all **dbx** subcommands associated with the specified event can be removed by using the **all** flag. The *EventNumber* parameter specifies the event from which the **dbx** subcommands are to be removed.

Flags:

all Removes all the **dbx** subcommands associated with the specified event.

Examples:

1. To remove all the dbx subcommands from event number 2, enter:
   ```
   delcmd 2 all
   ```
2. To remove **dbx** subcommand number 1 from event number 3, enter:
   ```
   delcmd 3 1
   ```
3. To remove **dbx** subcommands numbers 1 and 2 from event number 2, enter:
   ```
   delcmd 2 1 2
   ```

See the **addcmd** subcommand, **clear** subcommand, the **delete** subcommand, **disable** subcommand, **enable** subcommand, the **stop** subcommand, the **status** subcommand, and the **trace** subcommand. Also see Setting and Deleting Breakpoints in in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

delete Subcommand

delete { *Number* ... | **all** | **tskip** [**for $t***threadnumber*]}

The **delete** subcommand removes traces and stops from the application program and **tskip** counts for a thread. The traces and stops to be removed can be specified through the *Number* parameters, or all traces and stops can be removed by using the **all** flag. Use the **status** subcommand to display the numbers associated by the dbx debug program with a trace or stop.

The remaining **tskip** count, which was set using the **tskip** subcommand for a thread, can be deleted using the **tskip** flag. Use the **status** subcommand to display the remaining thread **tskip** counts. If no thread is specified, the current thread is used.

Flag

all	Removes all traces and stops.
for $t *threadnumber*	Specifies the thread number.

Examples

1. To remove all traces and stops from the application program, enter:
   ```
   delete all
   ```
2. To remove traces and stops for event number 4, enter:
   ```
   delete 4
   ```
3. To remove the **tskip** count for thread 3, enter:
   ```
   delete tskip for $t3
   ```
4. To remove the **tskip** count for the current thread, enter:
   ```
   delete tskip
   ```

See the **clear** subcommand, the **cleari** subcommand, the **status** subcommand, the **tskip** subcommand, and Setting and Deleting Breakpoints in in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

detach Subcommand

detach [*SignalNumber* | *SignalName*]

The **detach** subcommand continues the execution of the application program and exits the debug program. A signal can be specified either by:

- Name, using the *SignalName* parameter
- Number, using the *SignalNumber* parameter

  Signal names are not case sensitive and the **SIG** prefix is optional.

  If a signal is specified, the program continues as if it had received that signal. If no signal is specified, the program continues as if no stop had occurred.

Examples

1. To continue execution of the application and exit **dbx**, enter:
   ```
   detach
   ```
2. To exit **dbx** and continue execution of the application as though it received signal `SIGREQUEST`, enter:
   ```
   detach SIGREQUEST
   ```

See Using the dbx Debug Program.

disable Subcommand

disable { *Number* ... **all** }

The **disable** subcommand disables traces and stops associated with debug events. The traces and stops to be disabled can be specified through the *Number* parameters, or all traces and stops can be disabled by using the **all** flag. Use the **status** subcommand to display the event numbers associated by the dbx debug program with a trace or stop.

Flags:

all Removes all traces and stops.

Examples:

1. To disable all traces and stops from the application program, type:
   ```
   disable all
   ```
2. To disable traces and stops for event number `4`, type:
   ```
   disable 4
   ```

For more information, see “enable Subcommand” on page 25, “delete Subcommand” on page 21, “status Subcommand” on page 45.

Also, see Setting and Deleting Breakpoints in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

display memory Subcommand

{ *Address,Address*/ | *Address*/ [*Count*] } [*Mode*] [>*File*]

The **display memory** subcommand, which does not have a keyword to initiate the command, displays a portion of memory controlled by the following factors:

The range of memory displayed is controlled by specifying either:

- Two *Address* parameters, where all lines between those two addresses are displayed,
  OR
- One *Address* parameter where the display starts and a *Count* that determines the number of lines displayed from *Address.*

Specify symbolic addresses by preceding the name with an & (ampersand). Addresses can be expressions made up of other addresses and the operators + (plus sign), - (minus sign), and * (indirection). Any expression enclosed in parentheses is interpreted as an address.

- The format in which the memory is displayed is controlled by the *Mode* parameter. The default for the *Mode* parameter is the current mode. The initial value of *Mode* is **X**. The possible modes include:

b	Prints a byte in octal.
c	Prints a byte as a character.
d	Prints a short word in decimal.
D	Prints a long word in decimal.
Df	Prints a double-precision decimal float number.
DDf	Prints a quadruple-precision decimal float number.
f	Prints a single-precision real number.
g	Prints a double-precision real number.
h	Prints a byte in hexadecimal.
Hf	Prints a single-precision decimal float number.
i	Prints the machine instruction.
lld	Prints an 8-byte signed decimal number.
llu	Prints an 8-byte unsigned decimal number.
llx	Prints an 8-byte unsigned hexadecimal number.
llo	Prints an 8-byte unsigned octal number.
o	Prints a short word in octal
O	Prints a long word in octal.
q	Prints an extended-precision floating-point number.
s	Prints a string of characters terminated by a null byte.
x	Prints a short word in hexadecimal.
X	Prints a long word in hexadecimal.

Flag:

>*File*	Redirects output to the specified file.

Examples:

1. To display one long word of memory content in hexadecimal starting at the address `0x3fffe460`, enter:
   ```
   0x3fffe460 / X
   ```

2. To display two bytes of memory content as characters starting at the variable `y` address, enter:
   ```
   &y / 2c
   ```
3. To display the sixth through the eighth elements of the FORTRAN character string a_string, enter:
   ```
   &a_string + 5, &a_string + 7/c
   ```

See Examining Memory Addresses in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

down Subcommand

down [*Count*]

The **down** subcommand moves the current function down the stack *Count* number of levels. The current function is used for resolving names. The default for the *Count* parameter is one.

Examples:

1. To move one level down the stack, enter:
   ```
   down
   ```
2. To move three levels down the stack, enter:
   ```
   down 3
   ```

See the **up** subcommand, the **where** subcommand, and Displaying a Stack Trace in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

dump Subcommand

dump [*Procedure* | "**PATTERN**"] [>*File*]

The **dump** subcommand displays the names and values of all variables in the specified procedure or those that match with the specified pattern. If the *Procedure* parameter is a period (**.**), then all active variables are displayed. If neither the *Procedure* nor ″**PATTERN**″ parameter is specified, the current procedure is used. The ″**PATTERN**″ parameter is a wildcard expression with the `*`, `?`, and [] meta-characters. When ″**PATTERN**″ is used, it displays all the matching symbols in the global space (from all the procedures). If the **>***File* flag is used, the output is redirected to the specified file.

Flags:

>*File* Redirects output to the specified file.

Examples:

1. To display names and values of variables in the current procedure, enter:
   ```
   dump
   ```
2. To display names and values of variables in the **add_count** procedure, enter:
   ```
   dump add_count
   ```
3. To display names and values of variables starting from the character `s`, enter:
   ```
   dump "s*"
   ```
4. To redirect names and values of variables in the current procedure to the **var.list** file, enter:
   ```
   dump > var.list
   ```

See Displaying and Modifying Variables in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

edit Subcommand

edit [*Procedure* | *File*]

The **edit** subcommand invokes an editor on the specified file. The file may be specified through the *File* parameter or by specifying the *Procedure* parameter, where the editor is invoked on the file containing that procedure. If no file is specified, the editor is invoked on the current source file. The default is the **vi** editor. Override the default by resetting the **EDITOR** environment variable to the name of the desired editor.

Examples:

1. To start an editor on the current source file, enter:
   ```
   edit
   ```
2. To start an editor on the `main.c` file, enter:
   ```
   edit main.c
   ```
3. To start an editor on the file containing the `do_count()` procedure, enter:
   ```
   edit do_count
   ```

See the **list** subcommand, the **vi** or **vedit** command. Also, see Changing the Current File or Procedure and Displaying the Current File in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

enable Subcommand

enable { *Number* ... **all** }

The **enable** subcommand enables traces and stops associated with debug events. The traces and stops to be enabled can be specified through the *Number* parameters, or all traces and stops can be enabled by using the **all** flag. Use the **status** subcommand to display the event numbers associated by the dbx debug program with a trace or stop.

Flags:

all Removes all traces and stops.

Examples:

1. To enable all traces and stops from the application program, type:
   ```
   enable all
   ```
2. To enable traces and stops for event number `4`, type:
   ```
   enable 4
   ```

For more information, see “disable Subcommand” on page 22, “delete Subcommand” on page 21, “status Subcommand” on page 45.

Also, see Setting and Deleting Breakpoints in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

fd Subcommand

fd [**raw**] [*start* [*end*]]

The **fd** subcommand displays file descriptor information. Using the **raw** option causes output to be displayed in raw hex format. Other optional arguments include *start* and *end* indices. If no index is given, then information about all available file descriptors is displayed. Use of one index displays a single file descriptor; two an inclusive range.

Examples:

1. To view information on all file descriptors in hex, type:
   ```
   fd raw
   ```
2. To view information on file descriptors in the range of 3 to 5, type:
   ```
   fd 3 5
   ```

file Subcommand

file [*File*]

The **file** subcommand changes the current source file to the file specified by the *File* parameter; it does not write to that file. The *File* parameter can specify a full path name to the file. If the *File* parameter does not specify a path, the **dbx** program tries to find the file by searching the use path. If the *File* parameter is not specified, the **file** subcommand displays the name of the current source file. The **file** subcommand also displays the full or relative path name of the file if the path is known.

Examples:

1. To change the current source file to the `main.c` file, enter:
   ```
   file main.c
   ```
2. To display the name of the current source file, enter:
   ```
   file
   ```

See the **func** subcommand. Also, see Changing the Current File or Procedure and Displaying the Current File in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

frame Subcommand

frame [*num*]

The **frame** subcommand changes the current function to the function corresponding to the specified stack frame number *num*. The current function is used for resolving names. The numbering of the stack frames starts from the currently active function's stack frame (the function frame that is currently active is always numbered `0`). If there are *n* frames, the frame of the **main** function will be numbered `n-1`. When no frame number is specified, information about the function associated with the current frame is displayed.

Examples:

1. To move to frame number `2`, enter:
   ```
   frame 2
   ```
2. To display the current function on the stack, enter:
   ```
   frame
   ```

See the **up** and **down** subcommands. Also, see Changing the Current File or Procedure and Displaying a Stack Trace in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

func Subcommand

func [*Procedure*]

The **func** subcommand changes the current function to the procedure or function specified by the *Procedure* parameter. If the *Procedure* parameter is not specified, the default current function is displayed. Changing the current function implicitly changes the current source file to the file containing the new function; the current scope used for name resolution is also changed.

Examples:

1. To change the current function to the `do_count` procedure, enter:

```
func do_count
```

2. To display the name of the current function, enter:

```
func
```

See the **file** subcommand. Also, see Changing the Current File or Procedure in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

goto Subcommand

goto *SourceLine*

The **goto** subcommand causes the specified source line to be run next. Normally, the source line must be in the same function as the current source line. To override this restriction, use the **set** subcommand with the **$unsafegoto** flag.

Example: To change the next line to be executed to line 6, enter:

```
goto 6
```

See the **cont** subcommand, the **gotoi** subcommand, and the **set** subcommand.

gotoi Subcommand

gotoi *Address*

The **gotoi** subcommand changes the program counter address to the address specified by the *Address* parameter.

Example: To change the program counter address to address `0x100002b4`, enter:

```
gotoi 0x100002b4
```

See the **goto** subcommand.

handler Subcommand

handler { **atfork** | **cancel_cleanup** [**all** | *pthread id*] }

The **handler** subcommand displays information about atfork or cancellation cleanup handlers registered using **pthread_atfork**, and **pthread_cleanup_push**, respectively. Using the *atfork* option, the names of routines registered as `pre`, `parent` and `child` atfork handlers are displayed (with their respective arguments in the case of non-posix compliant atfork handlers). The **cancel_cleanup** option causes display of all registered cancellation cleanup handlers, with an optional *pthread id* parameter specifying a particular pthread, or **all** specifying all pthreads. If none is given, then the cancellation cleanup handlers for the current pthread are displayed, if there are any.

Examples:

1. To view information on all registered atfork handlers, type:

```
handler atfork
```

2. To view information about any registered cancellation cleanup handlers for the current pthread, type:

```
handler cancel_cleanup
```

3. To view information about any registered cancellation cleanup handlers for the pthread object referred to as `$t2`, type:

```
handler cancel_cleanup 2
```

help Subcommand

help [*Subcommand* | *Topic*]

The **help** subcommand displays help information for **dbx** subcommands or topics, depending upon the parameter you specify. Entering the **help** subcommand with the *Subcommand* parameter displays the syntax statement and description of the specified subcommand. Entering the **help** subcommand with the *Topic* parameter displays a detailed description of the specified topic. You do not need to provide the entire topic string with the **help** subcommand. The **dbx** program can recognize the topic if you provide a substring starting from the beginning of the topic. The following topics are available:

startup	Lists **dbx** startup options.
execution	Lists **dbx** subcommands related to program execution.
breakpoints	Lists **dbx** subcommands related to breakpoints and traces.
files	Lists **dbx** subcommands for accessing source files.
data	Lists **dbx** subcommands for accessing program variables and data.
machine	Lists descriptions of **dbx** subcommands for machine-level debugging.
environment	Lists **dbx** subcommands for setting **dbx** configuration and environment.
threads	Lists **dbx** subcommands for accessing thread-related objects.
expressions	Describes **dbx** expression syntax and operators.
scope	Describes how **dbx** resolves names from different scopes.
set_variables	Lists **dbx** debug variables with a usage description.
usage	Lists common **dbx** subcommands with brief descriptions.

Examples:

1. To list all available **dbx** subcommands and topics, enter:
   ```
   help
   ```
2. To display the description of the **dbx** subcommand **list**, enter:
   ```
   help list
   ```
3. To display the description of the **dbx** topic **set_variables**, enter:
   ```
   help set_variables
   ```

ignore Subcommand

ignore [*SignalNumber* | *SignalName*]

The **ignore** subcommand stops the trapping of a specified signal before that signal is sent to the application program. This subcommand is useful when the application program being debugged handles signals such as interrupts.

The signal to be trapped can be specified by:

- Number, with the *SignalNumber* parameter
- Name, with the *SignalName* parameter

Signal names are not case sensitive. The **SIG** prefix is optional.

If neither the *SignalNumber* nor the *SignalName* parameter is specified, all signals except the **SIGHUP**, **SIGCLD**, **SIGALRM**, and **SIGKILL** signals are trapped by default. The **dbx** debug program cannot ignore the **SIGTRAP** signal if it comes from a process outside of the debugger. If no arguments are specified, the list of currently ignored signals will be displayed.

Example: To cause **dbx** to ignore alarm clock time-out signals sent to the application program, enter:

```
ignore alrm
```

See the **catch** subcommand. Also, see Handling Signals in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

kthread Subcommand

kthread [**raw**] [**info** | **ru**] [*tid*]

The **kthread** subcommand displays information about kernel threads. Using the **raw** option causes all output to be displayed in hex, regardless of whether it could be displayed in a more human-readable format. Using no arguments, summary information on all kernel threads is printed. Supplying a numeric thread ID causes dbx to show information on a single thread. The **info** option produces more detailed output about a thread, from the user thread structure. Use of the **ru** option displays the `ti_ru` data member, which contains resource usage information.

For more information about user threads, see “thread Subcommand” on page 50.

Examples:

1. To find information on the thread that is currently running, you must first obtain information about all threads by typing the following on the command line:

   ```
   kthread
   ```

   Threads that were running (or runnable) just before dbx stopped the process are marked with an asterisk. Choose the correct thread ID based on the output and type:

   ```
   kthread info tid
   ```

2. To view resource information in hex about all threads, type:

   ```
   kthread raw ru
   ```

list Subcommand

list [*Procedure* | *SourceLine-Expression* [**,***SourceLine-Expression*]]

The **list** subcommand displays a specified number of lines of the source file. The number of lines displayed are specified in one of two ways:

- By specifying a procedure using the *Procedure* parameter.

In this case, the **list** subcommand displays lines starting a few lines before the beginning of the specified procedure and until the list window is filled.

- By specifying a starting and ending source line number using the *SourceLine-Expression* parameter.

The *SourceLine-Expression* parameter should consist of a valid line number followed by an optional + (plus sign), or - (minus sign), and an integer. In addition, a *SourceLine* of $ (dollar sign) may be used to denote the current line number; a *SourceLine* of @ (at sign) may be used to denote the next line number to be listed.

All lines from the first line number specified to the second line number specified, inclusive, are then displayed.

If the second source line is omitted, the first line is printed only.

If the **list** subcommand is used without parameters, the number of lines specified by $listwindow are printed, beginning with the current source line.

To change the number of lines to list by default, set the special debug program variable, *$listwindow*, to the number of lines you want. Initially, *$listwindow* is set to 10.

Examples:

1. To list the lines `1` through `10` in the current file, enter:

   ```
   list 1,10
   ```

2. To list 10, or $listwindow, lines around the main procedure, enter:
   ```
   list main
   ```
3. To list 11 lines around the current line, enter:
   ```
   list $-5,$+5
   ```
4. You can use simple integer expressions involving addition and subtraction in *SourceLineExpression* expressions. For example:
   ```
   (dbx) list $
   4 {

   (dbx) list 5
   5 char i = '4';

   (dbx) list sub
   23 char *sub(s,a,k)
   24 int a;
   25 enum status k;  .  .  .

   (dbx) move
   25
   (dbx) list @ -2
   23 char *sub(s,a,k)
   ```

See the **edit** subcommand, the **listi** subcommand, and the **move** subcommand. Also, see Displaying the Current File in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

listi Subcommand

listi [*Procedure* | **at** *SourceLine* | *Address* [**,** *Address*]]

The **listi** subcommand displays a specified set of instructions from the source file. The instructions displayed are specified by:

- Providing the *Procedure* parameter, where the **listi** subcommand lists instructions from the beginning of the specified procedure until the list window is filled.
- Using the **at** *SourceLine* flag, where the **listi** subcommand displays instructions beginning at the specified source line and continuing until the list window is filled. The *SourceLine* variable can be specified as an integer or as a filename string followed by a : (colon) and an integer.
- Specifying a beginning and ending address using the *Address* parameters, where all instructions between the two addresses, inclusive, are displayed.

If the **listi** subcommand is used without flags or parameters, the next **$listwindow** instructions are displayed. To change the current size of the list window, use the **set $listwindow=***Value* subcommand.

Disassembly Modes: The **dbx** program can disassemble instructions for either the POWER family or PowerPC architecture. In the default mode, the **dbx** program displays the instructions for the architecture on which it is running.

The **$instructionset** and **$mnemonics** variables of the **set** subcommand for the **dbx** command allow you to override the default disassembly mode. For more information, see the **set** subcommand for the **dbx** command.

Flag:

at *SourceLine* Specifies a starting source line for the listing.

Examples:

1. To list the next 10, or $listwindow, instructions, enter:
   ```
   listi
   ```

2. To list the machine instructions beginning at source line `10`, enter:
   ```
   listi at 10
   ```
3. To list the machine instructions beginning at source line `5` in file `sample.c`, enter:
   ```
   listi at "sample.c":5
   ```
4. To list the instructions between addresses `0x10000400` and `0x10000420`, enter:
   ```
   listi 0x10000400, 0x10000420
   ```

See the **list** subcommand and the **set** subcommand. Also, see Debugging at the Machine Level with dbx in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

malloc Subcommand

malloc [> *File*]

The **malloc** subcommand with no options prints out a list of enabled options and allocation policies as well as a statistical summary of malloc usage since process startup.

malloc [**allocation** [{ *address* | *size* | *heap* | *pid* | *tid* | *time* } { *"<"* | *"=="* | *">"* *"!="*] } *Value*]] [> *File*]

The **allocation** option to the **malloc** subcommand displays a sorted list of all the allocations currently held by the process. Using an optional `attribute RELOP value` argument allows for a more narrow selection of active allocations.

malloc [**freespace** [{ *address* | *size* | *heap* } { *"<"* | *"=="* | *">"* | *"!="*] } *Value*]] [> *File*]

The **freespace** option to the **malloc** subcommand displays a sorted list of all the free space available in the process heap. Using an optional `attribute RELOP value` argument allows for a more narrow selection of free space nodes.

Flags:

> *File* Redirects output to the specified file.

For more information, see System Memory Allocation Using the malloc Subsystem in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

map Subcommand

map { [*Format*] [**entry** *ModuleNumber* [, *ModuleNumber*] | *Address* | *SymbolName*] [**for $t***threadnumber*] [**>** *File*] }

The **map** subcommand displays characteristics for loaded portions of the application. This information can include the module name, member name, text origin, text end, text length, data origin, data end, data length, TLS data origin, TLS data end, TLS data length, and file descriptor for each loaded module. The entries to be displayed can be specified in the following ways:

- By specifying a single entry using the *ModuleNumber* parameter.
- By specifying a range of entries using two comma-separated *ModuleNumber* parameters.
- By specifying an address to be resolved to a loaded module using the *Address* parameter.
- By specifying a symbol name to be resolved to a loaded module using the *SymbolName* parameter.

When called without one of the above specifications, the map subcommand displays information for all loaded portions of the application.

The *Format* argument specifies the output mode for the loaded module descriptions. The following list contains possible values for the *Format* argument:

abbr	Specifies the abbreviated output mode, which consists of a single line for each loaded module containing the entry number, module name, and optional member name for that module.
normal	Specifies the normal output mode, which consists of the entry number, module name, member name, text origin, text length, data origin, data length, and file descriptor for each loaded module. If the loaded module has TLS data, the TLS data origin and TLS data length are also displayed.
raw	Specifies the raw output mode, which consists of a single unformatted line for each module containing the following space-separated fields: entry number, module name with optional member name, text origin, text end, text length, data origin, data end, data length, and file descriptor. If the loaded module has TLS data, the TLS data origin, TLS data end, and TLS data length are also displayed.
verbose	Specifies the verbose output mode, which consists of the entry number, module name, member name, text origin, text end, text length, data origin, data end, data length, and file descriptor for each loaded module. If the loaded module has TLS data, the TLS data origin, TLS data end, and TLS data length are also displayed.

If no *Format* parameter is specified, DBX uses the value of the **$mapformat** internal variable. If no *Format* parameter is specified and **$mapformat** is unset, DBX displays loaded module information in normal mode.

The TLS data information of the specified thread is displayed if the loaded module has TLS data. If no thread is specified, the current thread is used.

Flags:

> *File*	Redirects output to the specified file.
entry *ModuleNumber* [, *ModuleNumber*]	Specifies the module or range of modules to be displayed.
for $t *threadnumber*	Specifies the thread number.

Examples:

1. To list all loaded modules in abbreviated mode, type the following:
   ```
   map abbr
   ```
2. To list loaded modules 3 through 5 in verbose mode, type the following:
   ```
   map verbose entry 3,5
   ```
3. To list the loaded module that contains address `0x20001000`, type the following:
   ```
   map 0x20001000
   ```
4. To list the loaded module that contains variable `foo`, type the following:
   ```
   map foo
   ```
5. To list the loaded modules in normal mode with TLS data information of the modules for the thread 2, type the following:
   ```
   map normal for $t2
   ```

For more information, see the **$mapformat** internal variable. See also, Debugging at the Machine Level with dbx in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

move Subcommand

move *SourceLine*

The **move** subcommand changes the next line to be displayed to the line specified by the *SourceLine* parameter. This subcommand changes the value of the @ (at sign) variable.

The *SourceLine* variable can be specified as an integer or as a file name string followed by a : (colon) and an integer.

Examples:

1. To change the next line to be listed to line 12, enter:

   ```
   move 12
   ```

2. To change the next line to be listed to line 5 in file sample.c, enter:

   ```
   move "sample.c":5
   ```

See the **list** subcommand. Also, see Displaying the Current File in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

multproc Subcommand

multproc [**on** | **parent** | **child** | **off**]

The **multproc** subcommand specifies the behavior of the **dbx** debug program when forked and exceed processes are created. The **on** flag is used to specify that a new **dbx** session will be created to debug the child path of a fork. The original **dbx** will continue to debug the parent path. The **parent** and **child** flags are used to specify a single path of a fork to follow. All flags except **off** enable **dbx** to follow an exceed process. The **off** flag disables multiprocess debugging. If no flags are specified, the **multproc** subcommand returns the current status of multiprocess debugging.

The **dbx** program uses Xwindows for multiprocess debugging. The **dbx** program opens as many windows as needed for multiprocessing. The title for each child window is the process ID (pid) of the child process. To switch between processes, use Xwindows handling techniques to activate the window where the dbx session is displayed. If the system does not have Xwindows support, a warning message is issued when the debugger forks, and the dbx program continues debugging only the parent process. Multiprocess debugging can also be unsuccessful for the following reasons:

- The **dbx** program is not running in an Xwindows environment.
- Xwindows is running but the **dbx** global **$xdisplay** variable is not set to a valid display name. The **$xdisplay** variable is initialized to the shell **DISPLAY** environment variable. The **set** *Name=Expression* **dbx** subcommand can be used to change the value of the display name.
- The **/tmp** directory does not allow read or write access to the debugging program. The **dbx** program requires a small amount of space in this directory when controlling an Xwindow environment.
- The system does not have enough resources to accommodate a new Xwindow.

If **$xdisplay** is set to a remote display, the user may not be able to see the newly created Xwindow. If the **$xdisplay** setting is not correct, Xwindows or other system resources report the cause of the failure.

The **dbx** program does not distinguish between different types of failures, but the following message is sent when the subcommand is not successful:

```
Warning: dbx subcommand multiproc fails. dbx
continued with multproc disabled.
```

The user-defined configuration of the newly created window can be defined under the **dbx_term** application name in the **.Xdefaults** file.

Flags:

on Enables multiprocess debugging.
off Disables multiprocess debugging.

Examples:

1. To check the current status of multiprocess debugging, enter:

   ```
   multproc
   ```

2. To enable multiprocess debugging, enter:

   ```
   multproc on
   ```

3. To disable multiprocess debugging, enter:

   ```
   multproc off
   ```

See the **screen** subcommand and the **fork** subroutine. Also, see Debugging Programs Involving Multiple Processes in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

mutex Subcommand

mutex [**lock** | **unlock** | **thnum** | **utid** | *MutexNumber* ...]

The **mutex** subcommand displays information about mutexes. If the *MutexNumber* parameter is given, the **mutex** subcommand displays information about the specified mutexes. If no flags or parameters are specified, the **mutex** subcommand displays information about all mutexes.

The information listed for each mutex is as follows:

`mutex`	Indicates the symbolic name of the mutex, in the form `$m`*MutexNumber*.
`type`	Indicates the type of the mutex: `non-rec` (non recursive), `recursi` (recursive) or `fast`.
`obj_addr`	Indicates the memory address of the mutex.
`lock`	Indicates the lock state of the mutex: `yes` if the mutex is locked, `no` if not.
`owner`	If the mutex is locked, indicates the symbolic name of the user thread which holds the mutex.
`blockers`	List the user threads which are blocked on this mutex variable.

> **Note:** The **print** subcommand of the **dbx** debug program recognizes symbolic mutex names, and can be used to display the status of the corresponding object.

Flags:

lock	Displays information about locked mutexes.
unlock	Displays information about unlocked mutexes.
thnum	Displays information about all the mutexes held by a particular thread.
utid	Displays information about all the mutexes held by a user thread whose user thread id matches the given user thread id.

Examples:

1. To display information about all mutexes, enter:

   ```
   mutex
   ```

2. To display information about all locked mutexes, enter:

   ```
   mutex  lock
   ```

3. To display information about mutexes number four, five and six enter:

   ```
   mutex 4 5 6
   ```

   The output is similar to:

   ```
   mutex   obj_addr         type     lock owner  blockers
   $m4    0x20003274        non-rec   no
   $m5    0x20003280        recursi   no
   $m6    0x2000328a        fast      no
   ```

4. To display information about all the mutexes held by thread 1, enter:

   ```
   mutex thnum 1
   ```

5. To display information about all the mutexes held by a thread whose user thread id is 0x0001, enter:

```
mutex utid 0x0001
```

See the **attribute** subcommand, the **condition** subcommand, the **print** subcommand, and the **thread** subcommand.

Also, see. Using Mutexes *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

next Subcommand

next [*Number*]

The **next** subcommand runs the application program up to the next source line. The *Number* parameter specifies the number of times the **next** subcommand runs. If the *Number* parameter is not specified, **next** runs once only.

If you use the **next** subcommand in a multithreaded application program, all the user threads run during the operation, but the program continues execution until the running thread reaches the specified source line. If you wish to step the running thread only, use the **set** subcommand to set the variable **$hold_next**. Setting this variable may result in deadlock since the running thread may wait for a lock held by one of the blocked threads.

Examples:

1. To continue execution up to the next source line, enter:
   ```
   next
   ```
2. To continue execution up to the third source line following the current source line, enter:
   ```
   next 3
   ```

See the **cont** subcommand, **goto** subcommand, **nexti** subcommand, **set** subcommand, and the **step** subcommand.

nexti Subcommand

nexti [*Number*]

The **nexti** subcommand runs the application program up to the next instruction. The *Number* parameter specifies the number of times the **nexti** subcommand will run. If the *Number* parameter is not specified, **nexti** runs once only.

If you use the **nexti** subcommand in a multithreaded application program, all the user threads run during the operation, but the program continues execution until the running thread reaches the specified machine instruction. If you wish to step the running thread only, use the **set** subcommand to set the variable **$hold_next**. Setting this variable may result in deadlock since the running thread may wait for a lock held by one of the blocked threads.

Examples:

1. To continue execution up to the next machine instruction, enter:
   ```
   nexti
   ```
2. To continue execution up to the third machine instruction following the current machine instruction, enter:
   ```
   nexti 3
   ```

See the **gotoi** subcommand, **next** subcommand, **set** subcommand, and **stepi** subcommand. Also, see Running a Program at the Machine Level in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

onceblock Subcommand

onceblock [**uninit** | **done**]

The **onceblock** subcommand displays information about blocks of initialization code registered using the **pthread_once** routine. With no arguments, information about all registered once blocks is shown. The optional **uninit** and **done** flags display only the once blocks that either have not, or have already executed, respectively, while supplying a numeric once ID displays information for a single once block.

Note: For the **onceblock** subcommand to work while debugging a live process, the environment variable AIXTHREAD_ONCE_DEBUG should be set equal to `ON`. Likewise, if debugging a core file, if said variable was not on when the process ran, the **onceblock** subcommand will not be able to obtain any information.

Examples:

1. To find out if any once blocks have not yet executed, type:

   ```
   onceblock uninit
   ```

plugin Subcommand

plugin [*Name* [*Command*]]

The plugin subcommand passes the command specified by the *Command* parameter to the plug-in specified by the *Name* parameter. If no parameters are specified, the names of all available plug-ins are displayed.

Examples:

1. To list all available plug-ins, type:

   ```
   plugin
   ```

2. To invoke the subcommand ″help″ of a plug-in named ″sample″, type:

   ```
   plugin sample help
   ```

3. To invoke the subcommand ″`interpret 0x20000688`″ of a plug-in named ″`xyz`″, type:

   ```
   plugin xyz interpret 0x20000688
   ```

See the **pluginload** subcommand and **pluginunload** subcommand. Also see Developing for the dbx Plug-in Framework in *AIX Version 6.1 General Programming Concepts*.

pluginload Subcommand

pluginload *File*

The **pluginload** subcommand loads the plug-in specified by the *File* parameter. The *File* parameter should specify a path to the plug-in.

Examples: To load the plug-in named ″`sample`″ located at ″`/home/user/dbx_plugins/libdbx_sample.so`″, type:

```
pluginload /home/user/dbx_plugins/libdbx_sample.so
```

See the **plugin** subcommand and **pluginunload** subcommand. Also see Developing for the dbx Plug-in Framework in *AIX Version 6.1 General Programming Concepts*.

pluginunload Subcommand

pluginunload *Name*

The **pluginunload** subcommand unloads the plug-in specified by the *Name* parameter.

Examples: To unload the plug-in named ″`sample`″, type:

```
pluginunload sample
```

See the **plugin** subcommand and **pluginload** subcommand. Also see Developing for the dbx Plug-in Framework in *AIX Version 6.1 General Programming Concepts*.

print Subcommand

print *Expression* ...

print *Procedure* ([*Parameters*])

The **print** subcommand does either of the following:

- Prints the value of a list of expressions, specified by the *Expression* parameters.
- Executes a procedure, specified by the *Procedure* parameter and prints the return value of that procedure. Parameters that are included are passed to the procedure.

Examples:

1. To display the value of `x` and the value of `y` shifted left two bits, enter:
   ```
   print x, y << 2
   ```
2. To display the value returned by calling the `sbrk` routine with an argument of 0, enter:
   ```
   print sbrk(0)
   ```

See the **assign** subcommand, the **call** subcommand, and the **set** subcommand.

proc Subcommand

proc [**raw**] [**cred** | **cru** | **ru** | **sigflags** | **signal**]

The **proc** subcommand displays information about the process. Usage of the **raw** option causes output to be displayed in raw hex, rather than interpreting values in a more human-readable fashion. Using the **proc** subcommand with no additional arguments outputs general information about the process, as is stored in the user process data structure. The **cred** option displays contents of the `pi_cred` data member, which describes the credentials of the process. The **cru** and **ru** options display data members `pi_cru` and `pi_ru` respectively, which contain resource usage information. The **sigflags** and **signal** options display information relating to the current signal status and registered signal handlers, as contained within the `pi_sigflags` and `pi_signal` data members.

Examples:

1. To view resource usage information for the current process (or core file) in raw hex, type:
   ```
   proc raw ru
   ```
2. To view signal handler information, type:
   ```
   proc signal
   ```

prompt Subcommand

prompt [*"String"*]

The **prompt** subcommand changes the **dbx** command prompt to the string specified by the *String* parameter.

Example: To change the prompt to `dbx>`, enter:

```
prompt "dbx>"
```

See Defining a New dbx Prompt in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

quit Subcommand

quit

The **quit** subcommand terminates all processes running in the **dbx** debugging session.

See the **detach** subcommand.

registers Subcommand

registers [>*File*]

The **registers** subcommand displays the values of general purpose registers, system control registers, floating-point registers, vector registers and the current instruction register.

- General purpose registers are denoted by the **$r***Number* variable, where the *Number* parameter indicates the number of the register.

  Note: The register value may be set to the **0xdeadbeef** hexadecimal value. The **0xdeadbeef** hexadecimal value is an initialization value assigned to general-purpose registers at process initialization.
- Floating point registers are denoted by the **$fr***Number* variable. By default, the floating-point registers are not displayed. To display the floating-point registers, use the **unset $noflregs dbx** subcommand.
- Vector registers are denoted by the **$vr***Number* variable. The **$novregs** internal variable controls whether vector registers are displayed. The **$novregs** variable is set by default, and vector registers are not displayed. When **$novregs** is not set, and vector registers are valid (either debugging a program on a vector capable processor, or analyzing a core file containing vector registers state), then all the vector registers are displayed (vr0–vr31, vrsave, vscr). Vector registers can also be referenced by type. For example, the **$vrNf** (float), **$vrNs** (short), and **$vrNc** (char) vector register variables can be used with the **print** and **assign** subcommands to display and set vector registers by type.

Note: The **registers** subcommand cannot display registers if the current thread is in kernel mode.

Flag:

>*File* Redirects output to the specified file.

See the **set** subcommand and the **unset** subcommand. Also, see Using Machine Registers in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

rerun Subcommand

rerun [*Arguments*] [**<** *File*] [**>** *File*] [**> >** *File*] [**2>** *File*] [**2> >** *File*] [**>&** *File*] [**> >&** *File*]

The **rerun** subcommand begins execution of the object file. The *Arguments* are passed as command line arguments. If the *Arguments* parameter is not specified, the arguments from the last **run** or **rerun** subcommand are reused.

Flags:

<*File*	Redirects input so that input is received from *File*.
>*File*	Redirects output to *File*.
> >*File*	Appends redirected output to *File*.
2>*File*	Redirects standard error to *File*.
2> >*File*	Appends redirected standard error to *File*.
>&*File*	Redirects output and standard error to *File*.
> >&*File*	Appends output and standard error to *File*.

See the **run** subcommand.

resource Subcommand

resource { **owner** | **waiter** } [**all** | *pthread id*]

The **resource** subcommand displays information about which resources pthreads currently hold or are waiting on. The first argument, which is required, indicates whether you are interested in viewing pthreads that own resources or are waiting for them. The second argument can be used to indicate all pthreads, or a specific one. If none is given, then only information relevant to the current pthread is displayed, if applicable.

Note: The **resource** subcommand is only useful for debugging processes that run with several debugging environmental variables set to `ON`. These include AIXTHREAD_MUTEX_DEBUG, AIXTHREAD_COND_DEBUG, AIXTHREAD_RWLOCK_DEBUG, AIXTHREAD_READ_OWNER and AIXTHREAD_WAITLIST_DEBUG. If these variables are not turned on while debugging a live process, or were not on when a debugger core file was generated, the **resource** subcommand will be able to retrieve less information or none at all. Because use of these features can degrade performance, it is recommended that they only be activated for debugging purposes.

Examples:

1. To ascertain whether the current pthread holds any resources, type:
   ```
   resource owner
   ```
2. To view which resources any pthreads are waiting on, type:
   ```
   resource waiter all
   ```

return Subcommand

return [*Procedure*]

The **return** subcommand causes the application program to execute until a return to the procedure specified by the *Procedure* parameter is reached. If the *Procedure* parameter is not specified, execution ceases when the current procedure returns.

Examples:

1. To continue execution to the calling routine, enter:
   ```
   return
   ```
2. To continue execution to the `main` procedure, enter:
   ```
   return main
   ```

rwlock Subcommand

rwlock [read | write | *RwlockNumber*....]

The **rwlock** subcommand displays information about rwlocks. If the *RwlockNumber* parameter is given, the **rwlock** subcommand displays information about the specified rwlocks. If no flags or parameters are specified, the **rwlock** subcommand displays information about all rwlocks.

The information for each **rwlock** is as follows:

`rwl`	Indicates the symbolic name of the rwlock, in the form $rw *RwlockNumber*.
`flag_value`	Indicates the flag value.
`owner`	Indicates the owner of the rwlock
`status`	Indicates who is holding the rwlock. The values are read (if held by reader), write (if held by writer), free (if free).
`wsleep[#]`	Indicates threads blocking in write. # indicates the total number of threads blocking in write.
`rsleep[#]`	Indicates threads blocking in read. # indicates the total number of threads blocking in read.

Note: The **print** subcommand of the dbx debug program recognizes symbolic rwlock names, and can be used to display the status of the corresponding object

Flags:

read Displays information about all rwlocks whose status is in read mode.
write Displays information about all rwlocks whose status is in write mode.

Examples:

1. To display information about all rwlocks, enter:

   ```
   rwlock
   ```

   The output is similar to:

   ```
   rwl     flag_value    owner status
   $rwl        1         $t1    write
           rsleeps[     0]:
           wsleeps[     0]:
   ```

2. To display information about all rwlocks in write mode:

   ```
   rwlock write
   ```

   The output is similar to:

   ```
   rwl     flag_value    owner status
   $rwl        1         $t1    write
           rsleeps[     0]:
           wsleeps[     0]:
   ```

See the **attribute** subcommand, the **condition** subcommand, **mutex** subcommand, the **print** subcommand, and the **thread** subcommand

run Subcommand

run [*Arguments*] [**<***File*] [**>***File*] [**> >***File*] [**2>***File*] [**2> >***File*] [**>&***File*] [**> >&***File*]

The **run** subcommand starts the object file. The *Arguments* are passed as command line arguments.

Flags:

<*File*	Redirects input so that input is received from *File*.
>*File*	Redirects output to *File*.
2>*File*	Redirects standard error to *File*.
> >*File*	Appends redirected output to *File*.
2> >*File*	Appends redirected standard error to *File*.
>&*File*	Redirects output and standard error to *File*.
> >&*File*	Appends output and standard error to *File*.

Example: To run the application with the arguments `blue` and `12`, enter:

```
run blue 12
```

See the **rerun** subcommand.

screen Subcommand

screen

The **screen** subcommand opens an Xwindow for the **dbx** command interaction. You continue to operate in the window in which the process originated.

The **screen** subcommand must be run while the **dbx** debug program is running in an Xwindows environment. If the **screen** subcommand is issued in a non-Xwindow environment, the **dbx** program displays a warning message and resumes debugging as if the **screen** subcommand had not been given. The **screen** subcommand can also be unsuccessful in the following situations:

- The **dbx** program is not running in an Xwindows environment.
- Xwindows is running but the **dbx** global **$xdisplay** variable is not set to a valid display name. The **$xdisplay** variable is initialized to the **DISPLAY** environment variable. The **dbx** subcommand **set** *Name=Expression* changes the value of the display name.
- Xwindows is running, but the **TERM** environment variable is not set to a valid command name to invoke a new window.
- The **/tmp** directory does not allow read or write access to the program. The **dbx** program requires a small amount of space in this directory when the screen command is executed.
- System does not have enough resources to accommodate a new Xwindow.

The **dbx** program does not distinguish between different types of failures, but the program does send the following message:

```
Warning: dbx subcommand screen fails. dbx
continues.
```

If **$xdisplay** is set to a remote display, the user may not be able to see the newly created Xwindow. If the **$xdisplay** setting is not correct, Xwindows or other system resources report the problem.

The user-defined configuration of the newly created window can be defined under the **dbx_term** application name in the **.Xdefaults** file.

Example: To open an Xwindow for **dbx** command interaction, enter:

```
screen
```

See Separating dbx Output From Program Output in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs* and AIXwindows Overview, in *AIX Version 6.1 AIXwindows Programming Guide*.

set Subcommand

set [*Variable=Expression*]

The **set** subcommand defines a value for the **dbx** debug program variable. The value is specified by the *Expression* parameter; the program variable is specified by the *Variable* parameter. The name of the variable should not conflict with names in the program being debugged. A variable is expanded to the corresponding expression within other commands. If the **set** subcommand is used without arguments, the variables currently set are displayed.

The following variables are set with the **set** subcommand:

$catchbp	Catches breakpoints during the execution of the next command.
$deferevents	Turns on the deferred events feature.
$expandunions	Displays values for each part of variant records or unions.
$frame	Uses the stack frame pointed to by the address designated by the value of **$frame** for doing stack traces and accessing local variables.
$hexchars	Prints characters as hexadecimal values.
$hexin	Interprets addresses in hexadecimal.
$hexints	Prints integers as hexadecimal values.

$hexstrings Prints character pointers in hexadecimal.

$hold_next Holds all threads except the running thread during the **cont**, **next**, **nexti**, and **step** subcommands. Setting this variable may result in deadlock since the running thread may wait for a lock held by one of the blocked threads.

$ignoreifhandler Does not stop when your program receives a signal which has a registered handler.

$ignoreload Does not stop when your program performs the **load**, **unload**, or **loadbind** subroutine.

$ignorenonbptrap Does not stop when your program encounters a non-breakpoint trap instruction and has a registered **SIGTRAP** handler.

$instructionset Overrides the default disassembly mode. The following list contains possible values for the *Expression* parameter:

″**default**″
: Specifies the architecture on which the **dbx** program is running.

″**com**″ Specifies the instruction set for the common intersection mode of the PowerPC and POWER family architectures. The **dbx** program defaults to POWER-based mnemonics.

″**pwr**″ Specifies the instruction set and mnemonics for the POWER family architecture.

″**pwrx**″ Specifies the instruction set and mnemonics for the POWER2 implementation of the POWER family architecture for AIX 5.1 and earlier.

″**pwr6**″ Specifies the instruction set and mnemonics for the POWER6 implementation of the PowerPC architecture.

″**601**″ Specifies the instruction set and mnemonics for the PowerPC 601 RISC Microprocessor for AIX 5.1 and earlier.

″**603**″ Specifies the instruction set and mnemonics for the PowerPC 603 RISC Microprocessor for AIX 5.1 and earlier.

″**604**″ Specifies the instruction set and mnemonics for the PowerPC 604 RISC Microprocessor.

″**970**″ Specifies the instruction set and mnemonics for the PowerPC 970 microprocessor.

″**ppc**″ Specifies the instruction set and mnemonics defined in the POWER-based architecture, excluding the optional instructions. These instructions are available in all POWER-based implementations except the PowerPC 601 RISC Microprocessor in AIX 5.11 and earlier.

″**any**″ Specifies any valid POWER-based or POWER family instruction. For instruction sets that overlap, the default is the POWER-based mnemonics.

If no value is set for the *Expression* parameter, the **dbx** program uses the default disassembly mode.

$java When set, also sets the following variables, placing **dbx** in a mode to debug Java™ applications. When unset, also unsets the following variables:

$ignorenonbptrap
: Suppresses notification of trap instructions generated by the Java Just-In-Time (JIT) compiler.

$listwindow Specifies the number of lines to list around a function and the number to list when the **list** subcommand is used without parameters. The default is 10 lines.

$mapaddrs Starts mapping addresses. Unsetting **$mapaddrs** stops address mapping.

$mapformat Specifies the default output mode for the **map** subcommand.

″**abbr**″ Specifies the abbreviated output mode, which consists of a single line for each loaded module containing the entry number, module name, and optional member name for that module.

″**normal**″
Specifies the normal output mode, which consists of the entry number, module name, member name, text origin, text length, data origin, data length, and file descriptor for each loaded module. If the loaded module has TLS data, the TLS data origin and TLS data length are also displayed.

″**raw**″ Specifies the raw output mode, which consists of a single unformatted line for each module containing the following space-separated fields: entry number, module name with optional member name, text origin, text end, text length, data origin, data end, data length, and file descriptor. If the loaded module has TLS data, the TLS data origin, TLS data end, and TLS data length are also displayed.

″**verbose**″
Specifies the verbose output mode, which consists of the entry number, module name, member name, text origin, text end, text length, data origin, data end, data length, and file descriptor for each loaded module. If the loaded module has TLS data, the TLS data origin, TLS data end, and TLS data length are also displayed.

If no value is set for the *Expression* parameter, the dbx program will use the ″normal″ output mode.

$mnemonics Changes the set of mnemonics to be used by the **dbx** program when disassembling.

″**default**″
Specifies the mnemonics that most closely match the specified instruction set.

″**pwr**″ Specifies the mnemonics for the POWER family architecture.

″**ppc**″ Specifies the mnemonics defined in the POWER-based architecture book, excluding the optional instructions.

If no value is set for the *Expression* parameter, the **dbx** program will use the mnemonics that most closely match the specified instruction set.

$noargs Omits arguments from subcommands, such as *where*, *up*, *down*, and *dump*.

$noflregs Omits the display of floating-point registers from the **registers** subcommand.

$novregs Omits the display of vector registers from the **registers** subcommand.

$octin Interprets addresses in octal.

$octints Prints integers in octal.

$pretty Displays complex C and C++ data structure (struts, unions, arrays) values in a *pretty printed* format in conjunction with the **print** subcommand.

″**on**″ Specifies pretty printing with each value on its own line and with indentation to represent the static scope of each value.

″**verbose**″
Specifies pretty printing with each value on its own line and with qualified names to represent the static scope of each value. A qualified name consists of a dot-separated list of the outer blocks with which the value is associated.

″**off**″ Specifies pretty printing off. This is the default.

$repeat Repeats the previous command if no command was entered.

$sigblock Blocks signals to your program.

$stack_details Displays the frame number and the register set for each active function or procedure displayed by the **where** subcommand.

$stepignore Controls how the **dbx** command behaves when the **step**/**tstep** subcommand runs on a source line that calls another routine for which no debugging information is available. This variable enables the **step**/**tstep** subcommand to step over large routines for which no debugging information is available. The following list contains possible values for the *Expression* parameter:

″**function**″
Performs the function of the **next**/**tnext** subcommand for the **dbx** command. This is the default value.

″**module**″
Performs the function of the **next**/**tnext** subcommand if the function is in a load module for which no debug information is available (such as a system library).

″**none**″ Performs the function of the **stepi**/**tstepi** subcommand for the **dbx** command in the background until it reaches an instruction for which source information is available. At that point **dbx** will display where execution has stopped.

$thcomp When **$thcomp** is set, the information displayed by the thread command **th-** is shown in a compressed format.

$unsafeassign Turns off strict type checking between the two sides of an **assign** statement. Even if the **$unsafeassign** variable is set, the two sides of an **assign** statement may not contain storage types of different sizes.

$unsafebounds Turns off subscript checking on arrays.

$unsafecall Turns off strict type checking for arguments to subroutines or function calls.

$unsafegoto Turns off the **goto** subcommand destination checking.

$vardim Specifies the dimension length to use when printing arrays with unknown bounds. The default value is 10.

$xdisplay Specifies the display name for Xwindows, for use with the multproc subcommand or the screen subcommand. The default is the value of the shell **DISPLAY** variable.

The **$unsafe** variables limit the usefulness of the **dbx** debug program in detecting errors.

Examples:

1. To change the default number of lines to be listed to `20`, enter:
   ```
   set $listwindow=20
   ```
2. To disable type checking on the `assign` subcommand, enter:
   ```
   set $unsafeassign
   ```
3. To disassemble machine instructions for the PowerPC 601 RISC Microprocessor for AIX 5.1 and earlier, enter:
   ```
   set $instructionset="601"
   ```

See the **unset** subcommand. Also, see Changing Print Output with Special Debug Program Variables in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

set edit [vi, emacs] or set -o [vi, emacs] Subcommand

The **set** subcommand with the **-o** or **edit** option may be used to turn on one of the line edit modes. If the **set-o vi** or **set edit vi** command is given, you are placed in the input mode of the *vi* line editor. If the **set -o emacs** or **set edit emacs** command is given, you are placed in the input mode of the *emacs* line editor.

Example:

1. To turn on the vi line editor, enter:
   ```
   set-o vi
   ```
   or
   ```
   set edit vi
   ```

sh Subcommand

sh [*Command*]

The **sh** subcommand passes the command specified by the *Command* parameter to the shell for execution. The **SHELL** environment variable determines which shell is used. The default is the **sh** shell. If no argument is specified, control is transferred to the shell.

Examples:

1. To run the `ls` command, enter:
   ```
   sh ls
   ```
2. To escape to a shell, enter:
   ```
   sh
   ```
3. To use the `SHELL` environment variable, enter:
   ```
   sh echo $SHELL
   ```

See Running Shell Commands from dbx in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

skip Subcommand

skip [*Number*]

The **skip** subcommand continues execution of the application program from the current stopping point. A number of breakpoints equal to the value of the *Number* parameter are skipped and execution then ceases when the next breakpoint is reached or when the program finishes. If the *Number* parameter is not specified, it defaults to a value of one.

Example: To continue execution until the second breakpoint is encountered, enter:

```
skip 1
```

Also see the **cont** subcommand.

source Subcommand

source *File*

The **source** subcommand reads **dbx** subcommands from the file specified by the *File* parameter.

Example: To read the **dbx** subcommands in the `cmdfile` file, enter:

```
source cmdfile
```

See Reading dbx Subcommands from a File in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

status Subcommand

status [**more**] [>*File*]

The **status** subcommand displays all user-defined breakpoints, tracepoints, and watchpoints, in addition to the remaining thread **tskip** counts (set by using the **tskip** subcommand). If the more parameter is specified, the **status** subcommand also displays the **dbx** subcommands associated with the breakpoints, tracepoints, and watchpoints. The **status** subcommand lists enabled events with square brackets ([]) surrounding the event number, disabled events with periods (..) surrounding the event number, and deferred events with angle brackets (<>) surrounding the event number.

The > flag sends the output of the **status** subcommand to a file specified in the *File* parameter.

Flag:

>*File* Redirects output to *File*.

Examples:

1. To display all user-defined breakpoints, tracepoints, and watchpoints, as well as the remaining thread **tskip** counts, type:

   ```
   status
   ```

   The output is similar to:

   ```
   [1] stop at 13
   [2] stop at 14
   .3. stop at 15
   .4. stop at 16
   [5] stop at 17
   <6> stop at 18 if g > 10
   <7> stop in func

   Remaining tskip counts:
   tskip 2 for $t1
   tskip 1 for $t5
   ```

   In the example output above, events 3 and 4 are disabled, and events 6 and 7 are deferred.
2. To display all user-defined breakpoints, tracepoints, and watchpoints with associated dbx subcommands, enter:

   ```
   status more
   ```

   The output is similar to the following:

   ```
   [1] stop at 13
       [1] where
   .2. stop at 14
       [1] where
       [2] registers
   <3> stop at 15 if g > 10
       [1] where; registers
   ```

See the **addcmd** subcommand, the **clear** subcommand, the **delete** subcommand, the **delcmd** subcommand, the **tskip** subcommand, the **stop** subcommand, and the **trace** subcommand for the **dbx** command.

Also, see Setting and Deleting Breakpoints in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

step Subcommand

step [*Number*]

The **step** subcommand runs source lines of the application program. Specify the number of lines to be executed with the *Number* parameter. If the *Number* parameter is omitted, it defaults to a value of 1.

If you use the **step** subcommand on a multithreaded application program, all the user threads run during the operation, but the program continues execution until the running thread reaches the specified source line. If you wish to step the running thread only, use the **set** subcommand to set the variable **$hold_next**. Setting this variable may result in deadlock since the running thread may wait for a lock held by one of the blocked threads.

Note: Use the **$stepignore** variable of the **set** subcommand to control the behavior of the **step** subcommand. The **$stepignore** variable enables the **step** subcommand to step over large routines for which no debugging information is available.

Examples:

1. To continue execution for one source line, enter:
   ```
   step
   ```
2. To continue execution for five source lines, enter:
   ```
   step 5
   ```
3. To prevent the **dbx** program from single-stepping the **printf** function, as illustrated in the following example code:
   ```
   60 printf ("hello world \n");
   ```
   enter:
   ```
   set $stepignore="function"; step
   ```

See the **cont** subcommand, the **goto** subcommand, the **next** subcommand, the **set** subcommand, and the **stepi** subcommand.

stepi Subcommand

stepi [*Number*]

The **stepi** subcommand runs instructions of the application program. Specify the number of instructions to be executed in the *Number* parameter. If the *Number* parameter is omitted, it defaults to one.

If used on a multithreaded application program, the **stepi** subcommand steps the running thread only. All other user threads remain stopped.

Examples:

1. To continue execution for one machine instruction, enter:
   ```
   stepi
   ```
2. To continue execution for 5 machine instructions, enter:
   ```
   stepi 5
   ```

See the **gotoi** subcommand, the **nexti** subcommand, and the **step** subcommand.

stop Subcommand

stop { [*Variable*] [**at** *SourceLine* | **in** *Procedure* | **on load** [*″ModuleName″*]] [**if** *Condition*]}

The **stop** subcommand halts the application program when certain conditions are fulfilled. The program is stopped when:

- The *Condition* is true when the **if** *Condition* flag is used.
- The *Procedure* is called if the **in** *Procedure* flag is used.
- The *Variable* is changed if the *Variable* parameter is specified.
- The *SourceLine* line number is reached if the **at** *SourceLine* flag is used.

  The *SourceLine* variable can be specified as an integer or as a file name string followed by a : (colon) and an integer.
- The *ModuleName* loaded module is loaded or unloaded if the **on load** flag is used and the *ModuleName* parameter is specified.

  The optional *ModuleName* variable can be specified as a single module name, or as a module name paired with a member name in the format:

ModuleName(*MemberName*)

- Any loaded module is loaded or unloaded if the on load flag is used and the *ModuleName* parameter is not specified.

After any of these commands, the **dbx** debug program responds with a message reporting the event it has built as a result of your command. The message includes the event ID associated with your breakpoint along with an interpretation of your command. The syntax of the interpretation might not be exactly the same as your command. For example:

```
stop in main
[1] stop in main
stop at 19 if x == 3
[2] stop at "hello.c":19 if x = 3
stop in func
<3> stop in func
stop g
<4> stop g
```

The numbers in square brackets ([]) are the event identifiers associated with the breakpoints. The **dbx** debug program associates event numbers with each **stop** subcommand. When the program is halted as the result of one of the events, the event identifier is displayed along with the current line to show which event caused the program to stop. The numbers in angle brackets (<>) are the event identifiers for the deferred events. A deferred event is an event without having any breakpoint, tracepoint, or watchpoint associated with it, and is created whenever the input command involves the symbols that are not currently loaded in the memory. A normal event displayed in square brackets ([]) is also converted into a deferred event whenever the corresponding module is unloaded. Whenever the module corresponding to the deferred event is loaded into the memory, the deferred event is converted into the normal event, and the corresponding breakpoint, tracepoint, or watchpoint is created. The events you create coexist with internal events created by **dbx**, so event numbers might not always be sequential.

Use the **status** subcommand to view these numbers. You can redirect output from **status** to a file. Use the **delete** or **clear** subcommand to turn the **stop** subcommand off, or use the **enable** or **disable** subcommands. Use the **addcmd** subcommand to add **dbx** subcommands to the specified event number and **delcmd** to delete the associated **dbx** subcommands from the specified event number.

In a multithreaded application program, all user threads are halted when any user thread hits a breakpoint. A breakpoint set on a source line or function will be hit by any user thread which executes the line or function, unless you specify conditions as shown in example 9 below. The following aliases specify the conditions automatically:

- **bfth**(*Function*, *ThreadNumber*)
- **blth**(*SourceLine*, *ThreadNumber*)

ThreadNumber is the number part of the symbolic thread name as reported by the **thread** subcommand (for example, 5 is the *ThreadNumber* for the thread name $t5). These aliases are actually macros which produce the expanded subcommands shown below:

```
stopi at &Function    if ($running_thread == ThreadNumber)
stop at SourceLine  if ($running_thread == ThreadNumber)
```

Flags:

at *SourceLine*	Specifies the line number.
if *Condition*	Specifies the condition, such as true.
in *Procedure*	Specifies the procedure to be called.
on load *ModuleName*	Specifies the loaded module to be monitored.

Examples:

1. To stop execution at the first statement in the `main` procedure, enter:

```
stop in main
```

2. To stop execution when the value of the `x` variable is changed on line `12` of the execution, enter:
   ```
   stop x at 12
   ```
3. To stop execution at line 5 in file `sample.c`, enter:
   ```
   stop at "sample.c":5
   ```
4. To check the value of `x` each time that **dbx** runs a subroutine within `func1`, enter:
   ```
   stop in func1 if x = 22
   ```
5. To check the value of `x` each time that **dbx** begins to run `func1`, enter:
   ```
   stopi at &func1 if x = 22
   ```
6. To stop the program when the value of *Variable* changes, enter:
   ```
   stop Variable
   ```
7. To stop the program whenever *Condition* evaluates to true, enter:
   ```
   stop if (x > y) and (x < 2000)
   ```
8. The following example shows how to display active events and remove them:
   ```
   status
   [1] stop in main
   [2] stop at "hello.c":19 if x = 3
   delete 1
   status
   [2] stop at "hello.c":19 if x = 3
   clear 19
   status
   (dbx)
   ```

   The **delete** command eliminates events by event identifier. The **clear** command deletes breakpoints by line number.
9. To place a breakpoint at the start of `func1` only when executed by thread `$t5`, enter one of the following equivalent commands:
   ```
   stopi at &func1 if ($running_thread == 5)
   ```

   or
   ```
   bfth(func1, 5)
   ```
10. To stop the program when any module is loaded or unloaded, enter:
    ```
    stop on load
    ```
11. To stop the program whenever module `Module` is loaded or unloaded, enter:
    ```
    stop on load "Module"
    ```
12. To stop the program whenever member `Member` of module `Module` is loaded or unloaded, enter:
    ```
    stop on load "Module(Member)"
    ```

See the **addcmd** subcommand, the **clear** subcommand, the **delete** subcommand, the **delcmd** subcommand, **disable** subcommand, **enable** subcommand, the **stopi** subcommand, and the **trace** subcommand. Also, see Setting and Deleting Breakpoints in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

stophwp Subcommand

stophwp *Address Size*

The **stophwp** subcommand sets a hardware watchpoint stop for the specified memory region. The program stops when the contents of the region change.

Notes:

1. The success of the **stophwp** subcommand is hardware dependent. This feature is available only on POWER630 and POWER4 onwards.

2. As a result of the hardware limitation of being able to set only a single watchpoint, an active watchpoint event acts as a conflict when attempting to create another hardware watchpoint event with **stophwp** and **tracehwp**. As such, the previous event must be deleted before creating a new one. Also, since the existence of an active software watchpoint (created by some invocations of the **stop** and **trace** subcommands) negate the performance gains of hardware watchpoints, these types of events also act as conflicts which must be deleted before creating a hardware watchpoint.

Example:

1. To stop the program when the contents of the 4 byte memory region starting at address 0x200004e8 change, enter:

```
stophwp 0x200004e8 4
```

See the **tracehwp** subcommand.

stopi Subcommand

stopi { [*Address*] [**at** *Address* | **in** *Procedure*] [**if** *Condition*]}

The **stopi** subcommand sets a stop at the specified location:

- With the **if** *Condition* flag, the program stops when the condition true is specified.
- With the *Address* parameter, the program stops when the contents of *Address* change.
- With the **at** *Address* flag, a stop is set at the specified address.
- With the **in** *Procedure* flag, the program stops when the *Procedure* is called.

Flags:

if *Condition*	Specifies the condition, such as true.
in *Procedure*	Specifies the procedure to be called.
at *Address*	Specifies the machine instruction address.

Examples:

1. To stop execution at address `0x100020f0`, enter:

```
stopi at 0x100020f0
```

2. To stop execution when the contents of address `0x100020f0` change, enter:

```
stopi 0x100020f0
```

3. To stop execution when the contents of address 0x100020f0 are changed by thread `$t1`, enter:

```
stopi 0x200020f0 if ($running_thread == 1)
```

See the **stop** subcommand . Also, see Debugging at the Machine Level with dbx in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

thread Subcommand

Display Selected Threads: **thread** { [**info**] [**-**] [*ThreadNumber* ...] } | **current** | **run** | **susp** | **term** | **wait**

Select an Individual Thread: **thread current** [-] *ThreadNumber*

Hold or Release Threads: **thread** { **hold** | **unhold** } [-] [*ThreadNumber* ...]

Help for the options displayed: **thread** { **help**}

The **thread** subcommand displays and controls user threads.

The first form of the **thread** subcommand can display information in two formats. If the **thread** subcommand is **th**, then the information displayed is in the first format. If the **thread** subcommand is **th -**, then the information displayed is in the second format. If no parameters are given, information about all user threads is displayed. If one or more *ThreadNumber* parameters are given, information about the corresponding user threads is displayed. When the **thread** subcommand displays threads, the current thread line is preceded by a **>**. If the running thread is not the same as the current thread, its line is preceded by a *****. The information displayed by the **thread** subcommand in both the formats is described below.

The information displayed by the **thread** subcommand in the first format is as follows:

`thread` Indicates the symbolic name of the user thread, in the form `$t`*ThreadNumber*.

`state-k` Indicates the state of the kernel thread (if the user thread is attached to a kernel thread). This can be `run`, `wait`, `susp`, or `term`, for running, waiting, suspended, or terminated.

`wchan` Indicates the event on which the kernel thread is waiting or sleeping (if the user thread is attached to a kernel thread).

`state-u` Indicates the state of the user thread. Possible states are `running`, `blocked`, or `terminated`.

`k-tid` Indicates the kernel thread identifier (if the user thread is attached to a kernel thread).

`mode` Indicates the mode (kernel or user) in which the user thread is stopped (if the user thread is attached to a kernel thread).

`held` Indicates whether the user thread has been held.

`scope` Indicates the contention scope of the user thread; this can be `sys` or `pro` for system or process contention scope.

`function` Indicates the name of the user thread function.

The information displayed by the **thread** subcommand in the second format is given below. By default, for the **thread** subcommand **th -**, the information is displayed in the long form.

`thread` Indicates the symbolic name of the user thread, in the form `$t`*ThreadNumber*.

Kernel thread related information

`tid` Indicates the user thread identifier (if the user thread is attached to a kernel thread).

`pri` Indicates the priority of the kernel thread.

`sched` Indicates the scheduling policy of the kernel thread. This can be fif, oth, rr, for fifo, other, or round robin scheduling policies.

`state` Indicates the state of the kernel thread (if the user thread is attached to a kernel thread). This can be run, wait, susp, or zomb, for running, waiting, suspended, or zombie.

User thread related information

`tid` Indicates the user thread identifier.

`pri` Indicates the priority of the userl thread.

`sched` Indicates the scheduling policy of the user thread. This can be fif, oth, rr, for fifo, other, or round robin scheduling policies.

`state` Indicates the state of the user thread. This can be running, creating, suspended, blocked, runnable, or terminated.

`state` Indicates the user state in hex.

`flags` Indicates the values for pthread flags in hex.

`wchan` Indicates the event on which the kernel thread is waiting or sleeping (if the user thread is attached to a kernel thread).

`mode` Indicates the mode (kernel or user) in which the user thread is stopped (if the user thread is attached to a kernel thread).

`held` Indicates whether the user thread has been held.

scope — Indicates the contention scope of the user thread; this can be sys or pro for system or process contention scope.

cancellation

pending — Indicates if cancellation is pending or not.

state — Indicates the mode and state of cancellation.

If the cancellation is not pending and the state and mode are enabled and deferred respectively, then it is represented by **ed**, if cancellation state and mode is enabled and asynchronous, then it is represented by **ea**, and if mode is not enabled, then it is represented by **d**.

If the cancellation is pending and the cancellation state and mode is enabled and deferred respectively, then it is represented by **ED**, if cancellation state and mode is enabled and asynchronous, then it is represented by **EA**, and if mode is not enabled, then it is represented by **D**.

joinable — Indicates whether the thread is joinable or not.

boosted — Indicates the boosted value of the thread.

function — Indicates the name of the user thread function.

cursig — Indicates the current signal value.

If the option set $thcomp is set, then the information is displayed in the compressed form as shown below.

```
m           mode           (k)ernel (u)ser
k           k-state        (r)unning (w)aiting (s)uspended (z)ombie
u           u-state        (r)unning (R)unnable (s)uspended (t)erminated

                           (b)locked (c)reating
h           held           (yes) (n)o
s           scope          (s)ystem (p)rocess
c           cancellation   not pending:  (e)nabled & (d)eferred,
                                         (e)nabled & (a)sync, (d)isabled
                           pending    :  (E)nabled & (D)eferred,
                                         (E)nabled & (A)sync, (D)isabled
j           joinable       (yes) (n)o
b           boosted        value of boosted field in pthread structure
plk         kernel thread  (oth)er  (fif)o  (rr)-> round-robin
            policy
plu         user thread    (oth)er  (fif)o  (rr)-> round-robin
            policy
prk         kernel thread  hex number
            policy
pru         user thread    hex number
            policy
k-tid                      kernel thread id in hex
u-tid                      pthread id in hex
fl                         value of flags field in pthread structure in hex
sta                        value of state field in pthread structure in hex
cs                         value of the current signal
wchan                      event for which thread is waiting
function                   function name
```

The second form of the **thread** subcommand is used to select the current thread. The **print**, **registers**, and **where** subcommands of the **dbx** debug program all work in the context of the current thread. The **registers** subcommand cannot display registers if the current thread is in kernel mode.

The third form of the **thread** subcommand is used to control thread execution. Threads can be held using the **hold** flag, or released using the **unhold** flag. A held thread will not be resumed until it is released.

Note: The **print** subcommand of the **dbx** debug program recognizes symbolic thread names, and can be used to display the status of the corresponding object.

Flags:

current If the *ThreadNumber* parameter is not given, displays the current thread. If the *ThreadNumber* parameter is given, selects the specified user thread as the current thread.

help Displays all the information about the thread options that are shown when **th -** command is used.

hold If the *ThreadNumber* parameter is not given, holds and displays all user threads. If one or more *ThreadNumber* parameters are given, holds and displays the specified user threads.

unhold If the *ThreadNumber* parameter is not given, releases and displays all previously held user threads. If one or more *ThreadNumber* parameters are given, releases and displays the specified user threads.

info If the *ThreadNumber* parameter is not given, displays a long format listing of all user threads. If one or more *ThreadNumber* parameters are given, displays a long format listing the specified user threads.

All the above flags take [-] option. If this option is given, then the thread information displayed is in the second format and in the long form unless the **set** `$thcomp` option is set.

run Displays threads which are in the `run` state.

susp Displays threads which are in the `susp` state.

term Displays threads which are in the `term` state.

wait Displays threads which are in the `wait` state.

Examples:

1. To display information about threads that are in the wait state, enter:

   ```
   thread  wait
   ```

   The output is similar to:

   ```
   thread  state-k   wchan state-u   k-tid mode held scope function
    $t1     wait            running   17381   u   no   pro  main
    $t3     wait            running    8169   u   no   pro  iothread
   ```

2. To display information about several given threads, enter:

   ```
   thread 1 3 4
   ```

   The output is similar to:

   ```
   thread  state-k   wchan state-u   k-tid mode held scope function
    $t1     wait           running   17381   u   no   pro  main
    $t3     wait           running    8169   u   no   pro  iothread
   >$t4     run            running    9669   u   no   pro  save_thr
   ```

3. To make thread 4 the current thread, enter:

   ```
   thread  current 4
   ```

4. To hold thread number 2, enter:

   ```
   thread  hold 2
   ```

5. To display information about threads that are in the wait state, in the second format, enter:

   ```
   thread wait -
   ```

   The output is similar to:

   ```
   thread m k u h s c  j b  kpl  upl  kpr upr k_tid   u_tid   fl  sta wchan  function
   *$t1   u r w n p ed y 0  oth  oth  61   1  0043e5  000001  51  004        main
    $t3   u r w n p ed y 0  oth  oth  61   1  001fe9  000102  51  004        iothread
   >$t4   u r r n p ed y 0  oth  oth  61   1  0025c5  000203  50  064        save_thr
   ```

6. To display information about several given threads in the second format, enter:

   ```
   thread - 1 2 3
   ```

   The output is similar to:

```
thread m k u h s c  j b  kpl  upl  kpr upr k_tid   u_tid   fl  sta wchan function
*$t1   u r w n p ed y 0  oth  oth  61   1  0043e5  000001  51  004       main
 $t3   u r w n p ed y 0  oth  oth  61   1  00fe9   000102  51  004       iothread
>$t4   u r r n p ed y 0  oth  oth  61   1  0025c5  000203  50  064       save_thr
```

See the **attribute** subcommand, the **condition** subcommand, the **mutex** subcommand, the **print** subcommand, the **registers** subcommand, and the **where** subcommand.

Also, see Creating Threads *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

tls Subcommand

tls map

The **tls** subcommand takes only one flag that it uses to display the TLS initialization template origin and length for each loaded TLS module.

tnext Subcommand

tnext [*Number*]

The **tnext** subcommand runs the running thread up to the next source line. The *Number* parameter specifies the number of times the **tnext** subcommand runs. If the *Number* parameter is not specified, **tnext** runs once only. This subcommand can only be invoked on system-scope threads.

All the threads are run during this operation. To catch breakpoints during this operation, set the `$catchbp` **dbx** variable. If the `$catchbp` variable has been set and a breakpoint is reached for another thread, the **tnext** subcommand will not be repeated for the remaining number of times.

Examples:

1. To continue execution of the running thread up to the next source line, enter:
   ```
   tnext
   ```
2. To continue execution of the running thread up to the third source line following the current source line, enter:
   ```
   tnext 3
   ```

See the **tnexti** subcommand. Also, see Debugging Programs Involving Multiple Threads in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

tnexti Subcommand

tnexti [*Number*]

The **tnexti** subcommand runs the running thread up to the next instruction. The *Number* parameter specifies the number of times the **tnexti** subcommand runs. If the *Number* parameter is not specified, **tnexti** runs once only. This subcommand can only be invoked on system-scope threads.

All the threads are run during this operation. To catch breakpoints during this operation, set the `$catchbp` **dbx** variable. If the `$catchbp` variable has been set and a breakpoint is reached for another thread, the **tnexti** subcommand will not be repeated for the remaining number of times.

Examples:

1. To continue execution of the running thread up to the next machine instruction, enter:
   ```
   tnexti
   ```

2. To continue execution of the running thread up to the third machine instruction following the current machine instruction, enter:

```
tnexti 3
```

See the **tnext** subcommand. Also, see Debugging Programs Involving Multiple Threads in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

trace Subcommand

trace [*SourceLine* | *Expression* **at** *SourceLine* | *Procedure* | [*Variable*] [**at** *SourceLine* | **in** *Procedure*] | **on load** *ModuleName*] [**if** *Condition*]

The **trace** subcommand prints tracing information for the specified procedure, function, source line, expression, or variable when the program runs. The *SourceLine* variable can be specified as an integer or as a file name string followed by a : (colon) and an integer. A condition can be specified. The **dbx** debug program associates a number with each **trace** subcommand. Use the **status** subcommand to view these numbers. Use the **delete** subcommand to turn tracing off. You can enable and disable traces using the **enable** and **disable** subcommands, respectively.

The **trace** subcommand can display tracing information when modules are loaded or unloaded by the debugged process. The optional *ModuleName* parameter can be specified as a single module name, or as a module name paired with a member name in the format:

ModuleName(*MemberName*)

If the **on load** flag is used without the *ModuleName* parameter, dbx will trace the load and unload of all modules.

By default, tracing is process based. In order to make a thread based trace, specify the thread in a condition as shown in example 8 below.

Flags:

at *SourceLine*	Specifies the source line where the expression being traced is found.
if *Condition*	Specifies a condition for the beginning of the trace. The trace begins only **if** *Condition* is true.
in *Procedure*	Specifies the procedure to use to find the procedure or variable being traced.
on load *ModuleName*	Specifies the load module to be monitored.

Examples:

1. To trace each call to the `printf` procedure, enter:

   ```
   trace printf
   ```

2. To trace each execution of line 22 in the `hello.c` file, enter:

   ```
   trace "hello.c":22
   ```

3. To trace changes to the `x` variable within the `main` procedure, enter:

   ```
   trace x in main
   ```

4. To trace the data address 0x2004000, enter:

   ```
   set $A=0x2004000
   trace $A
   ```

   Note: The **tracei** subcommand is designed to trace addresses.

5. You can restrict the printing of source lines to when the specified *Procedure* is active. You can also specify an optional *Condition* to control when trace information should be produced. For example:

```
(dbx) trace in sub2
[1] trace in sub2
(dbx) run
trace in hellosub.c:  8  printf("%s",s);
trace in hellosub.c:  9  i = '5';
trace in hellosub.c:  10  }
```

6. You can display a message each time a procedure is called or returned. When a procedure is called, the information includes passed parameters and the name of the calling routine. On a return, the information includes the return value from *Procedure*. For example:

```
(dbx) trace sub
[1] trace sub
(dbx) run
calling sub(s = "hello", a = -1, k = delete) from function main
returning "hello" from sub
```

7. You can print the value of *Expression* when the program reaches the specified source line. The lines number and file are printed, but the source line is not. For example:

```
(dbx) trace x*17 at "hellosub.c":8 if (x > 0)
[1] trace x*17 at "hellosub.c":8 if x > 0
(dbx) run
at line 8 in file "hellosub.c": x*17 = 51

(dbx) trace x
[1] trace x
initially (at line 4 in "hello.c"):  x = 0
after line 17 in "hello.c":  x = 3
```

8. To trace changes to the **x** variable made by thread `$t1`, enter:

```
(dbx) trace x if ($running_thread == 1)
```

9. To trace the load or unload of all modules, enter:

```
trace on load
```

10. To trace the load or unload of module `Module`, enter:

```
trace on load "Module"
```

11. To trace the load or unload of member `Member` in module `Module`, enter:

```
trace on load "Module(Member)"
```

Also, see the **tracei** subcommand.

tracehwp Subcommand

tracehwp *Address Size*

The **tracehwp** subcommand sets a hardware watchpoint stop for the specified memory region. The **dbx** debug program prints tracing information when the contents of the region change.

Notes:

1. The success of the **tracehwp** subcommand is hardware dependent. This feature is available only on POWER630 and POWER4 onwards.
2. As a result of the hardware limitation of being able to set only a single watchpoint, an active watchpoint event acts as a conflict when attempting to create another hardware watchpoint event with **stophwp** and **tracehwp**. As such, the previous event must be deleted before creating a new one. Also, since the existence of an active software watchpoint (created by some invocations of the **stop** and **trace** subcommands) negate the performance gains of hardware watchpoints, these types of events also act as conflicts which must be deleted before creating a hardware watchpoint.

Examples:

1. To trace each time the contents of the 4 byte memory region starting at address 0x200004e8 change, enter:

```
tracehwp 0x200004e8 4
```

See the **stophwp** subcommand.

tracei Subcommand

tracei [[*Address*] [**at** *Address* | **in** *Procedure*] | *Expression* **at** *Address*] [**if** *Condition*]

The **tracei** subcommand turns on tracing when:

- The contents of the address specified by the *Address* parameter change if the *Address* flag is included.
- The instruction **at** Address is run if the **at** *Address* parameter is specified.
- The procedure specified by *Procedure* is active if the **in** *Procedure* flag is included.
- The condition specified by the *Condition* parameter is true if the **if** *Condition* flag is included.

Flags:

at *Address*	Specifies an address. Tracing is enabled when the instruction at this address is run.
if *Condition*	Specifies a condition. Tracing is enabled when this condition is met.
in *Procedure*	Specifies a procedure. Tracing is enabled when this procedure is active.

Examples:

1. To trace each instruction executed, enter:
   ```
   tracei
   ```
2. To trace each time the instruction at address `0x100020f0` is executed, enter:
   ```
   tracei at 0x100020f0
   ```
3. To trace each time the contents of memory location `0x20004020` change while the `main` procedure is active, enter:
   ```
   tracei 0x20004020 in main
   ```
4. To trace each time the instruction at address 0x100020f0 is executed by thread $t4, enter:
   ```
   tracei at 0x100020f0 if ($running_thread == 4)
   ```

See the **trace** subcommand. Also, see Debugging at the Machine Level with dbx in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

tskip Subcommand

tskip [*Number*]

The **tskip** subcommand continues the execution of the running thread from the current stopping point. The number of thread-level breakpoints specified by the *Number* parameter will be skipped for the running thread. This subcommand can be invoked for system-scope threads only.

All the other threads are run during this operation, and all breakpoints and watchpoints specified by the user are caught. The execution can cease when any thread hits a breakpoint or watchpoint. Even though the execution started by **tskip** subcommand can stop because of an event for another thread, the **tskip** count specified for the previous thread will still be active and the number of thread-level breakpoints specified by the **tskip** count will be ignored for that thread when the process continues. When the thread ends, the **tskip** count associated with it will be deleted.

Use the **status** subcommand to view the remaining **tskip** count for the threads. Use the **delete** subcommand to delete the remaining **tskip** count for the threads.

Example: To continue execution until the second thread-level breakpoint is encountered starting from the current stopping point for the running thread, enter:

```
tskip 1
```

See the cont subcommand. Also, see Debugging Programs Involving Multiple Threads in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

tstep Subcommand

tstep [*Number*]

The **tstep** subcommand runs the specified number of source lines from the current source line for the running thread. The *Number* parameter specifies the number of times the **tstep** subcommand runs. If the *Number* parameter is not specified, **tstep** runs once only. This subcommand can only be invoked on system-scope threads.

All the threads are run during this operation. If `$hold_next` is set, all the threads except the running thread will be held.

Note: Use the `$stepignore` variable of the **set** subcommand to control the behavior of the **tstep** subcommand. The `$stepignore` variable enables the **tstep** subcommand to step over large routines for which no debugging information is available.

Examples:

1. To continue execution of the running thread up for one source line, enter:
   ```
   tstep
   ```
2. To continue execution of the running thread for five source lines, enter:
   ```
   tstep 5
   ```
3. To prevent the **dbx** program from single-stepping the **printf** function, as illustrated in the example code:
   ```
   60 printf ("hello world /n");
   ```
   enter:
   ```
   set $stepignore="function"; step
   ```

See the cont subcommand, the goto subcommand, **tnext** subcommand, the set subcommand, and the **tstepi** subcommand. Also, see Debugging Programs Involving Multiple Threads in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

tstepi Subcommand

tstepi [*Number*]

The **tstepi** subcommand runs the specified number of instructions from the current instruction for the running thread. The *Number* parameter specifies the number of times the **tstepi** subcommand runs. If the *Number* parameter is not specified, **tstepi** runs once only. This subcommand can only be invoked on system-scope threads.

All the threads are run during this operation. If `$hold_next` is set, all the threads except the running thread will be held.

Examples:

1. To continue execution of the running thread up for one machine instruction, enter:
   ```
   tstepi
   ```
2. To continue execution of the running thread for five machine instructions, enter:
   ```
   tstepi 5
   ```

See the gotoi subcommand, **tnexti** subcommand, and the **tstep** subcommand. Also, see Debugging Programs Involving Multiple Threads in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

tstop Subcommand

tstop { **in** *Procedure* | [*Variable*] **at** *SourceLine* [**if** *Condition*] } [**for $t***threadnumber*]

The **tstop** subcommand sets a source-level breakpoint stop for a thread and halts the application program when the specified thread reaches the breakpoint. The thread specified should exist at the same time as the creation of the event. The current thread is used if no thread has been specified. The specified thread is stopped when any of the following occurs:

- The **if** *Condition* flag is used, and the *Condition* is true.
- The **in** *Procedure* flag is used, and the *Procedure* is called.
- The **at** *SourceLine* flag is used, and the *SourceLine* line number is reached. The *SourceLine* variable can be specified as an integer or as a file name string followed by a colon (:) and an integer.

Thread-level breakpoints can be set on system scope threads only. When a thread-level and a process-level breakpoint are hit at the same time, both the breakpoints will be processed and the thread-level breakpoint will be reported. When the thread terminates, the events associated with it will be deleted.

Flags:

at *SourceLine*	Specifies the line number.
for $t *threadnumber*	Specifies the thread number.
if *Condition*	Specifies the condition (for example, `true`).
in *Procedure*	Specifies the procedure to be called.

Examples:

1. To stop execution at the first statement in the **func** procedure while running thread 2, enter:
   ```
   tstop in func for $t2
   ```
2. To stop execution of the current thread when the value of the x variable is changed on line 12 of the execution, enter:
   ```
   tstop x at 12
   ```

See the **ttrace** subcommand. Also, see Debugging Programs Involving Multiple Threads in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

tstophwp Subcommand

tstophwp *address size* [**for $t***threadnumber*]

The **tstophwp** subcommand sets a thread-level hardware watchpoint stop for the specified memory region. The program stops when the contents of the region changes while running the specified thread. The thread specified should exist at the same time as the creation of the event. The current thread is used if no thread has been specified. The thread-level watchpoint events can be set only for system-scope threads. When the thread terminates, the events associated with it will be deleted.

Notes:

1. The success of the **tstophwp** subcommand is hardware dependent. This feature is available only on POWER630 and POWER4 onwards.
2. As a result of the hardware limitation allowing only a single watchpoint to be set, an active thread watchpoint event acts as a conflict when attempting to create another hardware watchpoint event for

the same thread using **tstophwp** and **ttracehwp**. To avoid this, the previous event must be deleted before creating a new one. Because the existence of an active software watchpoint (created by some invocations of the **stop** and **trace** subcommands) can negate the performance gains of hardware watchpoints, these types of events must also be deleted before creating a hardware watchpoint to avoid conflicts.

3. When a process-level watchpoint exists, a thread having no thread-level watchpoint will watch the process watchpoint location. If a thread has a thread-level watchpoint, the thread will watch the thread watchpoint location.
4. A thread-level hardware watchpoint and a process-level hardware watchpoint can coexist and do not conflict with each other.
5. If a process-level and a thread-level watchpoint exist for the same address, the process-level watchpoint event will be reported.

Flags:

for $t *threadnumber* Specifies the thread number.

Example: To stop the program when thread 2 is running and the contents of the 4-byte memory region starting at address `0x200004e8` change, enter:

```
tstophwp 0x200004e8 4 for $t2
```

See the **ttracehwp** subcommand. Also, see Debugging Programs Involving Multiple Threads in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

tstopi Subcommand

tstopi { **in** *Procedure* | [*Address*] **at** *Address* [**if** *Condition*] } [**for $t***threadnumber*]

The **tstopi** subcommand sets a instruction-level breakpoint stop for a thread. The thread specified should exist at the same time as the creation of the event. The current thread is used if no thread has been specified. The specified thread is stopped when any of the following occurs:

- The **if** *Condition* flag is used, and the *Condition* is true.
- The **in** *Procedure* flag is used, and the *Procedure* is called.
- The **at** *Address* flag is used, and the *Address* is reached.

Thread-level breakpoints can be set on system scope threads only. When a thread-level and a process-level breakpoint are hit at the same time, both the breakpoints will be processed and the thread-level breakpoint will be reported. When the thread terminates, the events associated with it will be deleted.

Flags:

at *Address* Specifies the machine instruction address.
for $t *threadnumber* Specifies the thread number.
if *Condition* Specifies the condition.
in *Procedure* Specifies the procedure to be called.

Example:

1. To stop execution at address `0x100020f0` while running thread 2, enter:

   ```
   tstopi at 0x100020f0 for $t2
   ```

2. To stop execution when the **func** procedure is entered while running the current thread, enter:

   ```
   tstopi in func
   ```

See the **ttracei** subcommand. Also, see Debugging Programs Involving Multiple Threads in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

ttrace Subcommand

ttrace { [*Variable*] **at** *SourceLine* | *Procedure* } [**if** *Condition*] [**for $t***threadnumber*]

The **ttrace** subcommand prints tracing information when the specified thread runs for the specified procedure, function, source line, and variable. The *SourceLine* variable can be specified as an integer or as a file name string followed by a colon (:) and an integer. The **dbx** debug program associates a number with each **ttrace** subcommand. Use the **status** subcommand to view these numbers. Use the **delete** subcommand to turn tracing off. You can enable and disable traces using the **enable** and **disable** subcommands, respectively.

The current thread will be used if no thread has been specified. Thread-level trace can be set only for system-scope threads. The thread specified should exist at the same time as the creation of the event. When the thread ends, the events associated with it will be deleted.

Flags:

at *SourceLine*	Specifies the source line where the expression being traced is found.
for $t *threadnumber*	Specifies the thread number.
if *Condition*	Specifies a condition for the beginning of the trace. The trace begins only if *Condition* is true.
in *Procedure*	Specifies the procedure to find the procedure or variable being traced.

Examples:

1. To trace each call to the **printf** procedure while running thread 2, enter:
   ```
   ttrace printf for $t2
   ```
2. To trace each execution of line 22 in the **hello.c** file while the current thread is running, enter:
   ```
   ttrace "hello.c":22
   ```

See the **ttracei** subcommand. Also, see Debugging Programs Involving Multiple Threads in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

ttracei Subcommand

ttracei [*Address*] **at** *Address* [**if** *Condition*] } [**for $t***threadnumber*]

The **ttracei** subcommand turns on tracing for the specified thread when any of the following occurs:

- The **if** *Condition* flag is included, and the *Condition* is true.
- The **at** *Address* flag is specified, and the instruction at *Address* is run.

The current thread will be used if no thread has been specified. Thread-level trace can be set only for system-scope threads. The thread specified should exist at the time as the creation of the event. When the thread ends, the events associated with it will be deleted.

Flags:

at *Address*	Specifies an address. Tracing is enabled when the instruction at this address is run.
for $t *threadnumber*	Specifies the thread number.
if *Condition*	Specifies a condition. Tracing is enabled when this condition is met.

Example:

1. To trace each time the instruction at address `0x100020f0` is executed while thread 3 is running, enter:

```
tracei at 0x100020f0 for $t3
```

2. To trace each time the instruction at address `0x100020f0` is executed by the current thread, enter:
```
tracei at 0x100020f0
```

See the **ttrace** subcommand. Also, see Debugging Programs Involving Multiple Threads in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

ttracehwp Subcommand

ttracehwp *address size* [**for $t***threadnumber*]

The **ttracehwp** subcommand sets a thread-level hardware watchpoint trace for the specified memory region. The **dbx** debug program prints tracing information when the contents of the region change while running the specified thread. The thread specified should exist at the same time as the creation of the event. The current thread is used if no thread has been specified. The thread-level watchpoint events can be set only for system-scope threads. When the thread terminates, the events associated with it will be deleted.

Notes:

1. The success of the **ttracehwp** subcommand is hardware dependent. This feature is available only on POWER630 and POWER4 onwards.
2. As a result of the hardware limitation allowing only a single watchpoint to be set, an active thread watchpoint event acts as a conflict when attempting to create another hardware watchpoint event for the same thread using **tstophwp** and **ttracehwp**. To avoid this, the previous event must be deleted before creating a new one. Because the existence of an active software watchpoint (created by some invocations of the **stop** and **trace** subcommands) can negate the performance gains of hardware watchpoints, these types of events must also be deleted before creating a hardware watchpoint to avoid conflicts.
3. When a process-level watchpoint exists, a thread having no thread-level watchpoint will watch the process watchpoint location. If a thread has a thread-level watchpoint, the thread will watch the thread watchpoint location.
4. A thread-level hardware watchpoint and a process-level hardware watchpoint can coexist and do not conflict with each other.
5. If a process-level and a thread-level watchpoint exist for the same address, the process-level watchpoint event will be reported.

Flags:

for $t *threadnumber* Specifies the thread number.

Example: To trace each time the contents of the 4-byte memory region starting at address `0x200004e8` change while running thread 2, enter:
```
ttracehwp 0x200004e8 4 for $t2
```

See the **tstophwp** subcommand. Also, see Debugging Programs Involving Multiple Threads in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

unalias Subcommand

unalias *Name*

The **unalias** subcommand removes the alias specified by the *Name* parameter.

Example: To remove an alias named `printx`, enter:
```
unalias printx
```

See the **alias** subcommand. Also, see Creating Subcommand Aliases in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

unset Subcommand

unset *Name*

The **unset** subcommand deletes the **dbx** debug program variable associated with the name specified by the *Name* parameter.

Example: To delete the variable inhibiting the display of floating-point registers, enter:

```
unset $noflregs
```

See the **set** subcommand. Also, see Changing Print Output With Special Debugging Variables in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

up Subcommand

up [*Count*]

The **up** subcommand moves the current function up the stack *Count* number of levels. The current function is used for resolving names. The default for the *Count* parameter is one.

Examples:

1. To move the current function up the stack 2 levels, enter:

   ```
   up 2
   ```

2. To display the current function on the stack, enter:

   ```
   up 0
   ```

See the **down** subcommand. Also, see Changing the Current File or Procedure, Displaying a Stack Trace in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

use Subcommand

use [{ **+** | *Directory* | '['*RegularExpression* = *NewPath*']' } ...]

The **use** subcommand sets the list of directories to be searched and path mappings to be applied when the **dbx** debug program looks for source files. If the **use** subcommand is specified without arguments, the current list of directories to be searched and path mappings to be applied are displayed.

The **@** (at-sign) is a special directory that directs the **dbx** program to look at the full-path name information in the object file, if it exists. If you have a relative directory called @ to search, you should use **./@** in the search path.

The **use** subcommand uses the **+** (plus-sign) to add more directories or mappings to the list of directories to be searched. The + represents the current list of directories and mappings when specified as input to the **use** subcommand. To append a directory or mapping to the end of the current list, the + should be specified before the new directory or mapping. To prepend a directory to the beginning of the current list, the + should be specified after the new directory or mapping. If you have a directory named **+**, specify the full-path name for the directory (for example, **./+** or **/tmp/+**).

The **use** subcommand interprets strings enclosed in [and] (square brackets) which contain an = (equal-sign) as path mappings. These path mappings are used in conjunction with the special @ directory. They make it easier for the user to express source file locations in the case that entire directory structures of source files have been relocated after compilation.

The following rules apply when attempting to locate a source file during debugging:

- Directories in the list are evaluated in the order specified.
- Upon evaluation of a directory in the list, the directory is searched for the given file. If the file exists in the directory and is readable, this file is used.
- Upon evaluation of the special @ directory, when one or more path mappings have been specified, if the *RegularExpression* portion of a path mapping matches the first *n* characters of the file's full-path name information in the object file and the substitution of the *NewPath* portion of the path mapping yields a readable file, this file is used.
- Upon evaluation of the special @ directory, when either no path mappings have been specified or none match, the directory corresponding to the file's full-path name information is searched. If the file exists in the directory and is readable, this file is used.
- If more than one path mapping yields a readable file, the path mapping whose *RegularExpression* matches the most characters (1 ... n) of the file's full-path name information (that is, the most specific) is applied and the resulting file is used.
- If more than one path mapping yields a readable file and each path mapping has equal specificity, the path mapping nearest to the beginning of the list is applied and the resulting file is used.

Note: If the special @ directory is not a member of the list, any path mappings that may have been specified will be completely ignored.

Examples:

1. To change the list of directories to be searched to the current directory (.), the parent directory (..), and the **/tmp** directory, enter:

   ```
   use . .. /tmp
   ```

2. To change the list of directories to be searched to the current directory (.), the directory the source file was located in at compilation time (@), and the **../source** directory, enter:

   ```
   use . @ ../source
   ```

3. To add the **/tmp2** directory to the list of directories to be searched, enter:

   ```
   use + /tmp2
   ```

4. To add the **/tmp3** directory to the beginning of the list of directories to be searched, enter:

   ```
   use /tmp3 +
   ```

5. To express that source files whose full-path name information begins with **/home/developer** are now located under **/mnt**, enter:

   ```
   use + [/home/developer=/mnt]
   ```

6. To direct the dbx program to first look under **/latest** and then, if the file does not exist there, to look under **/stable** for files with full-path name information beginning with **/home/developer**, enter:

   ```
   use + [/home/developer=/latest] [/home/developer=/stable]
   ```

Also, see the **edit** subcommand and the **list** subcommand.

whatis Subcommand

whatis *Name*

The **whatis** subcommand displays the declaration of *Name*, where the *Name* parameter designates a variable, procedure, or function name, optionally qualified with a block name.

Note: Use the **whatis** subcommand only while running the **dbx** debug program.

Examples:

1. To display the declaration of the `x` variable, enter:

   ```
   whatis x
   ```

2. To display the declaration of the `main` procedure, enter:

   ```
   whatis main
   ```

3. To display the declaration of the `x` variable within the `main` function, enter:

   ```
   whatis main.x
   ```

4. To print the declaration of an enumeration, structure, or union tag (or the equivalent in Pascal), use `$$TagName`:

   ```
   (dbx) whatis $$status
   enum $$status { run, create, delete, suspend };
   ```

where Subcommand

where [*startframe endframe*] [**>***File*]

The **where** subcommand displays a list of active procedures and functions associated with the frame numbers *startframe* to *endframe*. The numbering of the stack frame starts from the currently active function's stack frame (which is always numbered `0`). If there are *n* frames, the frame of the **main** function will be numbered `n-1`. By using the **>***File* flag, the output of this subcommand can be redirected to the specified file.

Flag:

>*File* Redirects output to the specified file.

See the **frame** subcommand, **up** subcommand, and **down** subcommand. Also, see Displaying a Stack Trace in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

whereis Subcommand

whereis *Identifier*

The **whereis** subcommand displays the full qualifications of all the symbols whose names match the specified identifier. The order in which the symbols print is not significant.

Examples: To display the qualified names of all symbols named `x`, enter:

```
whereis x
```

Also, see the **which** subcommand.

which Subcommand

which *Identifier*

The **which** subcommand displays the full qualification of the given identifier. The full qualification consists of a list of the outer blocks with which the identifier is associated.

Examples: To display the full qualification of the `x` symbol, enter:

```
which x
```

See the **whereis** subcommand. Also. see Scoping of Names in in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

Files

a.out	Object file; contains object code.
core	Contains core dump.

.dbxinit Contains initial commands.

Related Information

The **adb** command, **cc** command.

The **ptrace** subroutine.

The **a.out** file, **core** file.

The dbx Symbolic Debug Program Overview and Using the dbx Debug Program in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

---

dc Command

Purpose

Provides an interactive desk calculator for doing arbitrary-precision integer arithmetic.

Syntax

dc [*File*]

Description

The **dc** command is an arbitrary-precision arithmetic calculator. The **dc** command takes its input from the *File* parameter or standard input until it reads an end-of-file character. Once the **dc** command receives the input, it evaluates the value and writes the evaluation to standard output. It operates on decimal integers, but you can specify an input base, an output base, and a number of fractional digits to be maintained. The **dc** command is structured as a stacking, reverse Polish notation calculation.

The **bc** command is a preprocessor for the **dc** command. It provides infix notation and a syntax similar to the C language, which implements functions and control structures for programs.

Subcommands

c	Cleans the stack: the **dc** command pops all values on the stack.
d	Duplicates the top value on the stack.
f	Displays all values on the stack.
i	Pops the top value on the stack and uses that value as the number radix for further input.
I	Pushes the input base on the top of the stack.
k	Pops the top of the stack and uses that value as a nonnegative scale factor. The appropriate number of places is displayed on output and is maintained during multiplication, division, and exponentiation. The interaction of scale factor, input base, and output base is reasonable if all are changed together.
l*x*	Pushes the value in the register represented by the *x* variable on the stack. The register represented by the *x* variable is not changed. All registers start with a value of 0.
L*x*	Treats the *x* variable as a stack and pops its top value onto the main stack.
o	Pops the top value on the stack and uses that value as the number radix for further output.
O	Pushes the output base on the top of the stack.
p	Displays the top value on the stack. The top value remains unchanged.
P	Interprets the top of the stack as a string, removes it, and displays it.

q	Exits the program. If the **dc** command is running a string, it pops the recursion level by two.
Q	Pops the top value on the stack and on the string execution level by that value.
s*x*	Pops the top of the stack and stores it in a register named *x*, where the *x* variable can be any character.
S*x*	Treats the *x* variable as a stack. It pops the top of the main stack and pushes that value onto the stack represented by the *x* variable.
v	Replaces the top element on the stack by its square root. Any existing fractional part of the option is taken into account, but otherwise, the scale factor is ignored.
x	Treats the top element of the stack as a character string and runs it as a string of **dc** commands.
X	Replaces the number on the top of the stack with its scale factor.
z	Pushes the number of elements in the stack onto the stack.
Z	Replaces the top number in the stack with the number of digits in that number.
Number	Pushes the specified value onto the stack. A *Number* is an unbroken string of the digits 0 through 9. To specify a negative number, precede it with _ (underscore). A number may contain a decimal point.
+ - / * % ^	Adds (+), subtracts (-), multiplies (*), divides (/), remainders (%), or exponentiates (^) the top two values on the stack. The **dc** command pops the top two entries off the stack and pushes the result on the stack in their place. The **dc** command ignores fractional parts of an exponent.
[*String*]	Puts the bracketed *String* parameter onto the top of the stack.
[**=** | **>** | **<**] *x*	Pops the top two elements of the stack and compares them. Evaluates the register represented by the *x* variable as if it obeys the stated relation.
!	Interprets the rest of the line as an operating system command.
?	Gets and runs a line of input.
;:	The **bc** command uses these characters for array operations.

Examples

1. To use the **dc** command as a calculator, enter:

```
You:    1 4 / p
System: 0
You:    1 k   [ Keep 1 decimal place ]s.
        1 4 / p
System: 0.2
You:    3 k   [ Keep 3 decimal places ]s.
        1 4 / p
System: 0.250
You:    16 63 5 / + p
System: 28.600
You:    16 63 5 + / p
System: 0.235
```

   Comments may be used in the **dc** command as in the example. Comments are enclosed in brackets and may be followed by **s.** (**[** *Comment***] s.**) is ignored by the **dc** command. Comments enclosed in brackets only are stored on the top of the stack.

   When you enter the **dc** command expressions directly from the keyboard, press Ctrl-D to end the **bc** command session and return to the shell command line.

2. To load and run a **dc** program file, enter:

```
You:    dc prog.dc
        5 lf x p [ 5 factorial ]s.
System: 120
You:    10 lf x p [ 10 factorial ]s.
System: 3628800
```

This entry interprets the **dc** program saved in the **prog.dc** program file, then reads from the workstation keyboard. The `lf x` evaluates the function stored in register `f`, which could be defined in the **prog.c** program file as:

```
[ f: compute the factorial of n ]s.
[ (n = the top of the stack) ]s.
[ If 1>n do b; If 1<n do r ]s.
 [d 1 >b d 1 <r] sf
[ Return f(n) = 1 ]s.
 [d - 1 +] sb
[ Return f(n) = n * f(n-1) ]s.
 [d 1 - lf x *] sr
```

You can create **dc** program files with any text editor or with the **-c** (compile) flag of the **bc** command. When you enter the **dc** command expressions directly from the keyboard, press Ctrl-D to end the **bc** command session and return to the shell command line.

Files

/usr/bin/dc	Contains the **dc** command.

Related Information

The **bc** command.

---

dd Command

Purpose

Converts and copies a file.

Syntax

dd [**bs=***BlockSize*][**cbs=***BlockSize*] [**conv=** [`ascii` | `block` | `ebcdic` | `ibm` | `unblock`] [`lcase` | `ucase`] [`iblock`] [`noerror`] [`swab`] [`sync`] [`oblock`] [`notrunc`]] [**count=***InputBlocks*] [**files=***InputFiles*] [**fskip=***SkipEOFs*] [**ibs=***InputBlockSize*] [**if=***InFile*] [**obs=***OutputBlockSize*][**of=***OutFile*] [**seek=***RecordNumber*] [**skip=***SkipInputBlocks*][**span=***yes*|*no*]

dd [*Option*=*Value*]

Description

The **dd** command reads the *InFile* parameter or standard input, does the specified conversions, then copies the converted data to the *OutFile* parameter or standard output. The input and output block size can be specified to take advantage of raw physical I/O.

Note: The term *Block* refers to the quantity of data read or written by the **dd** command in one operation and is not necessarily the same size as a disk block.

Where sizes are specified, a number of bytes is expected. A number ending with **w**, **b**, or **k** specifies multiplication by 2, 512, or 1024 respectively; a pair of numbers separated by an **x** or an ***** (asterisk) indicates a product. The count parameter expects the number of blocks, *not* the number of bytes, to be copied.

The character-set mappings associated with the **conv=ascii** and **conv=ebcdic** flags are complementary operations. These flags map between ASCII characters and the subset of EBCDIC characters found on most workstations and keypunches.

Use the **cbs** parameter value if specifying any of the **block**, **unblock**, **ascii**, **ebcdic**, or **ibm** conversions. If **unblock** or **ascii** parameters are specified, then the **dd** command performs a fixed-length to variable-length conversion. Otherwise it performs a conversion from variable-length to fixed-length. The **cbs** parameter determines the fixed-length.

Attention: If the **cbs** parameter value is specified smaller than the smallest input block, the converted block is truncated.

After it finishes, the **dd** command reports the number of whole and partial input and output blocks.

Notes:

1. Usually, you need only write access to the output file. However, when the output file is not on a direct-access device and you use the **seek** flag, you also need read access to the file.
2. The **dd** command inserts new-line characters only when converting with the **conv=ascii** or **conv=unblock** flags set; it pads only when converting with the **conv=ebcdic**, **conv=ibm**, or **conv=block** flags set.
3. Use the **backup**, **tar**, or **cpio** command instead of the **dd** command whenever possible to copy files to tape. These commands are designed for use with tape devices. For more information on using tape devices, see the **rmt** special file.
4. The block size values specified with the **bs**, **ibs** and **obs** flags must always be a multiple of the physical block size for the media being used.
5. When the **conv=sync** flag is specified, the **dd** command pads any partial input blocks with nulls. Thus, the **dd** command inserts nulls into the middle of the data stream if any of the reads do not receive a full block of data (as specified by the **ibs** flag). This is a common occurrence when reading from pipes.
6. If the **bs** flag is specified by itself and no conversions other than **sync**, **noerror** or **notrunc** are specified, then the data from each input block will be written as a separate output block; if the read returns less than a full block and **sync** is not specified, then the resulting output block will be the same size as the input block. If the **bs** flag is not specified, or a conversion other than **sync**, **noerror** or **notrunc** is specified, then the input will be processed and collected into fullsized output blocks until the end of input is reached.

Spanning across devices

The **dd** can be made to span across devices if the input file is larger than the output device physical size.

Note: Care has to be taken when specifying the block size *bs* as exact multiple of the physical size of the device because improper block size will result in data inconsistency, or overlap.

The spanning of **dd** across devices will not occur if either one of the InFile or the OutFile parameter is stdin or stdout.

Spanning will occur in such a way that **dd** will prompt for next device during write if the output device is full. During read from the input device, **dd** will prompt for next device if the data is completely read from the input device even when the device has not reached the end. In this case it would be required to press 'n' to quit.

Flags

bs=*BlockSize*	Specifies both the input and output block size, superseding the **ibs** and **obs** flags. The block size values specified with the **bs** flag must always be a multiple of the physical block size for the media being used.
cbs=*BlockSize*	Specifies the conversion block size for variable-length to fixed-length and fixed-length to variable-length conversions, such as **conv=block**.
count=*InputBlocks*	Copies only the number of input blocks specified by the *InputBlocks* variable.

conv= *Conversion*,....
Specifies one or more conversion options. Multiple conversions should be separated by commas. The following list describes the possible options:

ascii Converts EBCDIC to ASCII. This option is incompatible with the **ebcdic**, **ibm**, **block**, and **unblock** options.

block Converts variable-length records to fixed-length. The length is determined by the conversion block size (cbs). This option is incompatible with the **ascii**, **ebcdic**, **ibm**, and **unblock** options.

ebcdic Converts ASCII to standard EBCDIC. This option is incompatible with the **ascii**, **ibm**, **block**, and **unblock** options.

ibm Converts ASCII to an IBM version of EBCDIC. This option is incompatible with the **ascii**, **ebcdic**, **block**, and **unblock** options.

iblock, oblock
Minimize data loss resulting from a read or write error on direct access devices. If you specify the **iblock** variable and an error occurs during a block read (where the block size is 512 or the size specified by the **ibs=***InputBlockSize* variable), the **dd** command attempts to reread the data block in smaller size units. If the **dd** command can determine the sector size of the input device, it reads the damaged block one sector at a time. Otherwise, it reads it 512 bytes at a time. The input block size (**ibs**) must be a multiple of this retry size. This option contains data loss associated with a read error to a single sector. The **oblock** conversion works similarly on output.

lcase Makes all alphabetic characters lowercase.

noerror
Does not stop processing on an error.

notrunc
Does not truncate the output file. Instead, blocks not explicitly written to output are preserved.

ucase Makes all alphabetic characters uppercase.

swab Swaps every pair of bytes.

sync Pads every input block to the **ibs** value.

unblock
Converts fixed-length blocks to variable-length. The length is determined by the conversion block size (cbs). This option is incompatible with the **ascii**, **ebcdic**, **ibm**, and **block** options.

files=*InputFiles*
Copies the number of files specified by the *InputFiles* variable value of input files before ending (makes sense only where input is a magnetic tape or similar device).

fskip=*SkipEOFs*
Skips past the number of end-of-file characters specified by the *SkipEOFs* variable before starting to copy; this *SkipEOFs* variable is useful for positioning on multifile magnetic tapes.

ibs=*InputBlockSize*
Specifies the input-block size; the default is 512 bytes or one block. The block-size values specified with the **ibs** flag must always be a multiple of the physical block size for the media being used.

if=*InFile*
Specifies the input file name; standard input is the default.

obs=*OutputBlockSize*
Specifies the output-block size; the default is 512 bytes or one block. The block size values specified with the **obs** flag must always be a multiple of the physical block size for the media being used.

of=*OutFile*
Specifies the output file name; standard output is the default.

seek=*RecordNumber*
Seeks the record specified by the *RecordNumber* variable from the beginning of output file before copying.

skip=*SkipInputBlocks* Skips the specified *SkipInputBlocks* value of input blocks before starting to copy.

span=*yes|no* Allows spanning across devices if specified `yes` and works as default if specified as `no`. See Spanning Across Devices, for more information..

Exit Status

This command returns the following exit values:

0 The input file was copied successfully.
>0 An error occurred.

Examples

1. To convert an ASCII text file to EBCDIC, type:

   ```
   dd  if=text.ascii  of=text.ebcdic  conv=ebcdic
   ```

   This command converts the `text.ascii` file to EBCDIC representation, storing the EBCDIC version in the `text.ebcdic` file.

   Note: When you specify the **conv=ebcdic** parameter, the **dd** command converts the ASCII ^ (circumflex) character to an unused EBCDIC character (9A hexadecimal), and the ASCII ~ (tilde) to the EBCDIC ^ (NOT symbol).
2. To convert the variable-length record ASCII file **/etc/passwd** to a file of 132-byte fixed-length EBCDIC records, type:

   ```
   dd  if=/etc/passwd  cbs=132  conv=ebcdic  of=/tmp/passwd.ebcdic
   ```
3. To convert the 132-byte-per-record EBCDIC file to variable-length ASCII lines in lowercase, type:

   ```
   dd  if=/tmp/passwd.ebcdic  cbs=132  conv=ascii  of=/tmp/passwd.ascii
   ```
4. To convert the variable-length record ASCII file **/etc/passwd** to a file of 132-byte fixed-length records in the IBM version of EBCDIC, type:

   ```
   dd  if=/etc/passwd  cbs=132  conv=ibm  of=/tmp/passwd.ibm
   ```
5. To copy blocks from a tape with 1KB blocks to another tape using 2KB blocks, type:

   ```
   dd  if=/dev/rmt0  ibs=1024  obs=2048  of=/dev/rmt1
   ```
6. To use the **dd** command as a filter, type:

   ```
   ls -l | dd  conv=ucase
   ```

   This command displays a long listing of the current directory in uppercase.

   Note: The performance of the **dd** command and **cpio** command to the 9348 Magnetic Tape Unit Model 12 can be improved by changing the default block size. To change the block size, use the **chdev** command in the following way:

   ```
   chdev -l Device_name -a block_size=32k
   ```
7. To perform efficient transfers to 3.5-inch 1.4MB diskette using 36 blocks of 512 bytes, type:

   ```
   dd  if=Filename  of=/dev/rfd0  bs=36b  conv=sync
   ```

   This command writes the value of the *Filename* parameter to the diskette device a cylinder at a time. The `conv=sync` is required when reading from disk and when the file size is not a multiple of the

diskette block size. Do not try this if the input to the **dd** command is a pipe instead of a file, it will pad most of the input with nulls instead of just the last block.

8. To copy blocks from a input file with block size set to 720b blocks into a 1.44MB size diskette type:

   ```
   dd if=testfile of=/dev/fd0 bs=720b conv=sync
   ```

   Note: If the input file is larger than the physical size of the output device then dd will prompt you for another device.

9. To copy blocks from a input file with block size set to 32k blocks to a tape type:

   ```
   dd if=inputfile of=/dev/rmt0 bs=32k conv=sync
   ```

10. To copy blocks of data from tape to a file in the current directory with block size set to 32k blocks type as follows:

    ```
    dd if=/dev/rmt0 of=outfile bs=32k conv=sync
    ```

11. To copy blocks from an input file with block size set to 720b, onto a 1.44MB size diskette, enter:

    ```
    dd if=testfile of=/dev/fd0 bs=720b conv=sync span=yes
    ```

    Note: If the input file is larger than the physical size of the output device, then **dd** will prompt you for another device.

12. To copy blocks from an input file with block size set to 32k, to a tape, enter:

    ```
    dd if=inputfile of=/dev/rmt0 bs=32k conv=sync span=yes
    ```

13. To copy blocks of data from tape with block size set to 32k, to a file in the current directory, enter:

    ```
    dd if=dev/rmt0 of=outfile bs=32k conv=sync span=yes
    ```

Files

/usr/bin/dd	Contains the **dd** command.

Related Information

The **backup**, **cp**, **cpio**, **tar**, **tr** command.

The **rmt** special file.

The Backup method in*Operating system and device management* provides information on using backups and using memory devices.

The Files in *Operating system and device management* provides information on working with files.

---

defaultbrowser Command

Purpose

Launches the default web browser and optionally loads a specified URL.

Syntax

defaultbrowser [*URL* [**new-window**, **new-tab**]]

Description

The **defaultbrowser** command runs the browser launch command that is specified in the DEFAULT_BROWSER environment variable.

If a *URL* is given as an argument, it loads that URL into the browser. For this to work properly, the browser command must accept a URL as an argument.

The optional **new-window** and **new-tab** arguments can be used if the browser that is being launched is the Mozilla Web browser. Both arguments must always be specified with a URL. This URL will then be opened in a new browser window or a new tab. If the browser is not the Mozilla Web browser, these two arguments will be ignored.

The main purpose of the **defaultbrowser** command is to have applications use this command when they need to open a browser to display HTML documents or web-based applications. This way, a system administrator only needs to change the DEFAULT_BROWSER environment variable when a new browser is installed and all applications will automatically begin using the new browser.

The DEFAULT_BROWSER environment variable should be set to the command that would launch the desired browser. Include any arguments that must be included after the command to launch a specific URL address. For example, if the command to launch a browser and open a specific URL is `wonderbrowser -r URL`, then the DEFAULT_BROWSER environment variable would be set to equal `wonderbrowser -r`.

If the DEFAULT_BROWSER environment variable is not defined, then the **defaultbrowser** command runs the Mozilla Web browser if it is installed.

Examples

1. To launch the designated default browser and have it open to it's default home page, type:
   ```
   defaultbrowser
   ```
2. To launch the designated default browser and have it open to the URL http://machine/path/file.html, type:
   ```
   defaultbrowser http://machine/path/file.html
   ```
3. To launch the designated default browser and have it open the URL http://machine/path/file.html where if the default browser is Netscape, then the page is displayed in a window called **webpage**, type:
   ```
   defaultbrowser http://machine/path/file.html webpage
   ```
4. To launch the designated default browser and have it open the URL http://machine/path/file.html in a new browser window if the browser is the Mozilla Web browser, type:
   ```
   defaultbrowser http://machine/path/file.html new-window
   ```
5. To launch the designated default browser and have it open the URL http://machine/path/file.html in a new browser tab if the browser is the Mozilla Web browser, type:
   ```
   defaultbrowser http://machine/path/file.html new-tab
   ```

Files

/usr/bin/defaultbrowser	The **defaultbrowser** command

---

defif Method

Purpose

Defines a network interface in the configuration database.

Syntax

defif [**-c** *Class* **-s** *Subclass*] **-t** *Type*

Description

The **defif** method defines the specified instance of a network interface. It only defines interfaces for currently configured adapters. To define the specified instance, the **defif** method does the following:

1. Creates a customized interface instance in the configuration database.
2. Derives the logical name of the interface instance.
3. Retrieves the predefined attributes.
4. Updates the Customized Dependency object class to reflect dependencies of the defined interface instance.
5. Sets the status flag of the interface instance to **defined**.

Flags

-c *Class* Specifies the interface class to be defined. The valid value is **if**.

-s *Subclass* Specifies the subclass of interface to be defined. Valid values are:

TR	Token-ring
EN	Ethernet
SL	Slip
XT™	X.25
LO	Loopback

-t *Type* Specifies the type of interface to be defined. Valid values are:

tr	Token-ring
en	Ethernet
sl	Slip
ie3	IEEE 802.3 Ethernet
lo	Loopback
xt	X.25

Examples

To define a token-ring network interface instance, enter the method in the following format:

```
defif -t tr
```

Related Information

The **mkdev** command.

The **odm_run_method** subroutine.

TCP/IP network interfaces in *Networks and communication management*.

Object Data Manager (ODM) Overview for Programmers in *General Programming Concepts*.

Writing a Device Method in *Kernel Extensions and Device Support Programming Concepts*.

---

definet Method

Purpose

Defines an inet instance in the system configuration database.

Syntax

definet [**-c** *Class*]

Description

The **definet** method creates an object in the ODM configuration database specifying the customized attributes of the inet instance. It performs the following operations:

1. Creates a customized inet instance.
2. Sets the status flag of the inet instance to defined.

This method is called by the **mkdev** high-level command and is not meant to be issued on the command line.

> **Note:** The **definet** method is a programming tool and should not be executed from the command line.

Flags

-c *Class* Specifies the inet instance to be defined. The only valid value for the *Class* variable is **tcpip**.

Examples

To define the inet0 instance, issue the following method:

```
definet
```

Related Information

The **mkdev** command.

The **odm_run_method** subroutine.

Object Data Manager (ODM) Overview for Programmers in *General Programming Concepts*.

Writing a Device Method in *Kernel Extensions and Device Support Programming Concepts*.

---

defragfs Command

Purpose

Increases a file system's contiguous free space.

Syntax

defragfs [**-q** | **-r** | **-s**] { *Device* | *FileSystem* }

Description

The **defragfs** command increases a file system's contiguous free space by reorganizing allocations to be contiguous rather than scattered across the disk. The file system to be defragmented can be specified with the *Device* variable, which is the path name of the logical volume (for example, **/dev/hd4**). It can also be specified with the *FileSystem* variable, which is the mount point in the **/etc/filesystems** file.

The **defragfs** command is intended for fragmented and compressed file systems. However, you can use the **defragfs** command to increase contiguous free space in nonfragmented file systems.

You must mount the file system read-write for this command to run successfully. Using the **-q** flag, the **-r** flag or the **-s** flag generates a fragmentation report. These flags do not alter the file system.

The **defragfs** command is slow against a JFS2 file system with a snapshot due to the amount of data that must be copied into snapshot storage object. The **defragfs** command issues a warning message if there are snapshots. The **snapshot** command can be used to delete the snapshots and then used again to create a new snapshot after the **defragfs** command completes.

Flags

-q Reports the current state of the file system.

-r Reports the current state of the file system and the state that would result if the **defragfs** command is run without either the **-q**, **-r** or **-s** flag.

-s Reports the fragmentation in the file system. This option causes **defragfs** to pass through meta data in the file system which may result in degraded performance.

Output

On a JFS filesystem, the definitions for the messages reported by the defragfs command are as follows:

Number of free fragments
The number of free fragments in the file system.

Number of allocated fragments
The number of allocated fragments in the file system.

Number of free spaces shorter than a block
The number of free spaces within the file system that are shorter than a block. A free space is a set of contiguous fragments that are not allocated.

Number of free fragments in short free spaces
The total number of fragments in all the short free spaces. A short free space is one that is shorter than a block.

Number of fragments moved
The total number of fragments moved.

Number of logical blocks moved
The total number of logical blocks moved.

Number of allocation attempts
The number of times free fragments were reallocated.

Number of exact matches
The number of times the fragments that are moved would fit exactly in some free space.

Total number of fragments
The total number of fragments in the file system.

Number of fragments that may be migrated
The number of fragments that may be moved during defragmentation.

FileSystem filesystem is n percent fragmented
Shows to what extent the file system is fragmented in percentage.

On a JFS2 filesystem the definitions for the messages reported by the defragfs command are as follows:

Total allocation groups
The number of allocation groups in the file system. Allocation groups divide the space on a file system into chunks. Allocation groups allow JFS2 resource allocation policies to use well known methods for achieving good I/O performance.

Allocation groups defragmented
The number of allocation groups that were defragmented.

Allocation groups skipped - entirely free
The number of allocation groups that were skipped because they were entirely free.

Allocation groups skipped - too few free blocks
The number of allocation groups that were skipped because there were too few free blocks in them for reallocation.

Allocation groups skipped - contains a large contiguous free space
The number of allocation groups that were skipped because they contained a large contiguous free space which is not worth defragmenting.

Allocation groups are candidates for defragmenting
The number of allocation groups that are fit for defragmenting.

Average number of free runs in candidate allocation groups
The average number of free runs per allocation group, for allocation groups that are found fit for defragmentation. A free run is a contiguous set of blocks which are not allocated.

Total number of blocks
The total number of blocks in the file system.

Number of blocks that may be migrated
The number of blocks that may be moved during defragmentation.

FileSystem filesystem is n percent fragmented
Shows to what extent the file system is fragmented in percentage.

Security

Attention RBAC users and Trusted AIX users: This command can perform privileged operations. Only privileged users can run privileged operations. For more information about authorizations and privileges, see Privileged Command Database in *Security*. For a list of privileges and the authorizations associated with this command, see the **/etc/security/privcmds** database file.

Examples

1. To defragment the **/data1** file system located on the **/dev/lv00** logical volume, enter:
   ```
   defragfs /data1
   ```
2. To defragment the **/data1** file system by specifying its mount point, enter:
   ```
   defragfs /data1
   ```
3. To generate a report on the **/data1** file system that indicates its current status as well as its status after being defragmented, enter:
   ```
   defragfs  -r /data1
   ```
4. To generate a report on the fragmentation in the **/data1** file system, enter:
   ```
   defragfs -s /data1
   ```

Files

/etc/filesystems	Lists the known file systems and defines their characteristics.

Related Information

The **crfs** command, the **lsfs** command, the **mkfs** command.

JFS data compression, JFS fragments and Variable number of i-nodes in the *Operating system and device management* book.

defvsd Command

Purpose

Designates a node as either having or using a virtual shared disk.

Syntax

defvsd *logical_volume_name global_group_name vsd_name*

Description

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

Flags

–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 virtual shared disks have been reconfigured on that node. The specified nodes are also cast in.

Note: This option should be used only under direct guidance from IBM Service. It should never be used under normal circumstances.

–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 virtual shared disk network will be cast in.

Note: This option should be used only under direct guidance from IBM Service. It should never be used under normal circumstances.

–p Sets the level of 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. (See *RSCT for AIX 5L: Managing Shared Disks* for more information regarding tuning virtual shared disk performance.)

This value is the *buf_cnt* parameter on the **uphysio** call that the 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.

–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.

Notes:

1. Before using this flag, refer to the “Restrictions” section that follows.
2. This option should be used only under direct guidance from IBM Service. It should never be used under normal circumstances.

–t Lists the current routing table and mbuf headers cached by the virtual shared disk driver.

–T Clears or releases all cached routes.

–v *vsd_name* **...**
Resets the statistics in the number of read and write requests on the specified virtual shared disks.

–V Resets all the configured virtual shared disk's statistics in the number of read and write requests.

–C Resets the 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.

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

Notes:

1. Before using this flag, refer to the "Restrictions" section that follows.
2. This option should be used only under direct guidance from IBM Service. It should never be used under normal circumstances.

–M Sets the virtual shared disk maximum IP message size. This is the largest sized block of data the virtual shared disk sends over the network for an I/O request. This limit also affects local virtual shared disk I/O block size. The value is in bytes and must not be greater than the maximum transmission unit (MTU) size of the network. All nodes should use the same value. The recommended values are:

- 61440 (60KB) for a switch
- 8192 (8KB) for jumbo frame Ethernet
- 1024 (1KB) for 1500-byte MTU Ethernet

Parameters

logical_volume_name
Is the name of the logical volume you want to specify as a 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 a 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 virtual shared disk. This name must be unique within the RSCT peer domain, and, in order to avoid possible future naming conflicts, should also be unique across the overall cluster. 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 specify a *vsd_name* that is already the name of another device, the **cfgvsd** command will be unsuccessful for that virtual shared disk. This error 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).

Security

You must have **root** authority to run this command.

Restrictions

You must issue this command from a node that is online in the peer domain. To bring a peer domain online, use the **startrpdomain** command. To bring a particular node online in an existing peer domain, use the **startrpnode** command. For more information on creating and administering an RSCT peer domain, refer to *RSCT Administration Guide* .

Examples

1. The following example specifies that, on the globally accessible volume group **vg1n1**, the logical volume known as **lv1vg1n1** is used as a virtual shared disk named **vsd1vg1n1**.

   ```
   defvsd lv1vg1n1 vg1n1 vsd1vg1n1
   ```

Location

/opt/rsct/vsd/bin/defvsd

Related Information

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

---

deleteX11input Command

Purpose

Deletes an X11 input extension record from the ODM (Object Data Manager) database.

Syntax

deleteX11input *DeviceName ...*

Description

The **deleteX11input** command is used to delete an X11 input extension record from the ODM database. For each *DeviceName* specified, the ODM database finds as many instances of the object as possible. This command queries the user to verify whether to delete each specific device found. A partial name may be specified.

The command is a root or system user command. Its action fails with a permissions error if an unauthorized user attempts to delete a record.

Parameter

DeviceName Specifies the name of the X11 input extension device.

Error Codes

No DeviceName is found in ODM Database	No objects that match the specified pattern were found in the ODM database.
Usage: deleteX11input DeviceName	The user has not specified a device name.

Related Information

The **addX11input** command, **listX11input** command.

delta Command

Purpose

Creates a delta in a SCCS file.

Syntax

delta [**-r** *SID*] [**-s**] [**-n**] [**-g** *List*] [**-p**] [**-m** *ModificationRequestList*] [**-y** [*Comment*]] *File ...*

Description

The **delta** command introduces into the named Source Code Control System (SCCS) file any changes that were made to the file version retrieved by a **get -e** command.

The **delta** command reads the g-files that correspond to the specified file*s* (see the **get** command for a description of files created and used by SCCS) and creates a new delta. No line of a g-file can contain more than 512 characters.

If you specify a directory for the *File* value, the **delta** command performs the requested actions on all SCCS files within that directory that have been checked out previously for editing (that is, on all files with an **s.** prefix). If you specify a **-** (minus sign) in place of the *File* value, the **delta** command reads standard input and interprets each line as the name of an SCCS file. When the **delta** command reads standard input, you must supply the **-y** flag. You must also supply the **-m** flag if the **v** header flag is set. The **delta** command reads standard input until it reaches an end-of-file character.

> **Note:** Lines beginning with an SOH ASCII character (binary 001) cannot be placed in the SCCS file unless the SOH is quoted using a **** (backslash). SOH has special meaning to SCCS and causes an error.

Use of a **get** command on SCCS files, followed by the **delta** command on those same files, should be avoided when the **get** command generates a large amount of data. Instead, you should alternate the use of the **get** and **delta** commands.

The **delta** command saves the changes made to a particular version of an SCCS file. To use the **delta** command:

1. Use the **get -e** command to get an editable version of the file.
2. Edit that file.
3. Use the **delta** command to create a new version of the SCCS file.

The **delta** command prompts you for comments if the **-y** option is not specified. The comments apply to that particular delta and appear in the SCCS file header. The comments are not retrieved when you use the **get** command to get the delta and do not appear in the text of a retrieved file. Use comments to keep track of why a delta was created.

To see the comments, use an editor to look at the SCCS file, write the SCCS file to the display screen with the **cat** command, or print selected parts of the file to standard output using the **prs** command. Remember not to change the contents of the SCCS file directly. To change the delta comments, use the **cdc** command.

> **Note:** Do not use the **delta** command on a file if it contains expanded identification keywords. Read-only file versions replace keywords with text values. Using the **delta** command on a read-only file causes the keywords to be lost. To recover from this situation, remove the delta or edit the file again and replace the identification keywords.

The SCCS does not allow use of the **delta** command unless an editable copy of the file exists.

To prevent the loss of keywords, use the **admin** command with the **-f** flag to specify the **i** header flag. Afterwards, the absence of keywords in a file version will cause an error.

Flags

-g *List*
Specifies a list of SIDs (deltas) to be ignored when the **get** command creates the g-file. After you use this flag, the **get** command ignores the specified delta when it builds the g-file.

-m *ModificationRequestList*
If the SCCS file has the **v** header flag set, then a Modification Request (MR) number must be supplied as the reason for creating the new delta.

If you do not specify the **-m** flag, and the **v** header flag is set, the **delta** command reads MRs from standard input. If standard input is a workstation, the **delta** command prompts you for the MRs. The **delta** command continues to take input until it reads an end-of-file character. It always reads MRs before the comments (see the **-y** flag). You can use blanks, tab characters, or both to separate MRs in a list.

If the **v** header flag has a value, it is interpreted as the name of a program that validates the MR numbers. If the **delta** command returns a nonzero exit value from the MR validation program, the **delta** command assumes some of the MR numbers were invalid and stops running.

-n
Retains the g-file, which is normally removed at completion of the **delta** command processing.

-p
Writes to standard output (in the format of the **diff** command) the SCCS file differences before and after the delta is applied. See the **diff** command for an explanation of the format.

-r *SID*
Specifies which delta is to be created in the SCCS file. You must use this flag only if two or more outstanding **get -e** commands were done on the same SCCS file by the same person. The *SID* value can be either the SID specified on the **get** command line or the SID to be created (as reported by the **get** command.) An error results if the specified SID cannot be uniquely identified, or if an SID must be specified but it is not.

-s
Suppresses the information normally written to standard output on normal completion of the **delta** command.

-y[*Comment*]
Specifies text that describes the reason for making a delta. A null string is considered a valid *Comment* value. If your comment line includes special characters or blanks, the line must be enclosed in single or double quotation marks.

If you do not specify the **-y** flag, the **delta** command reads comments from standard input until it encounters a blank line or an end-of-file character.

For keyboard input, the **delta** command prompts for the comments. If the last character of a line is a \ (backslash), it is ignored. Comments must be no longer than 512 characters.

Exit Status

This command returns the following exit values:

0 Successful completion.
>0 An error occurred.

Examples

1. To record changes you have made to an SCCS file, enter:

   ```
   delta s.prog.c
   ```

   This adds a delta to the SCCS file `s.prog.c`, recording the changes made by editing `prog.c`. The `delta` program then asks you for a comment that summarizes the changes you made. Enter the comment, and then enter an end-of-file character or press the return key twice to indicate that you have finished the comment.
2. To record the changes you have made to an SCCS file with a brief descriptive comment, enter:

   ```
   delta -y "This delta contains the payroll function" s.prog.c
   ```

Files

/usr/bin/delta	Contains the **delta** command.

Related Information

The **admin** command, **cat** command, **cdc** command, **diff** command, **get** command, **prs** command, **rmdel** command, **sccsdiff** command, and **sccshelp** command.

The **sccsfile** file format.

List of SCCS Commands in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

Source Code Control System (SCCS) Overview in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

---

deroff Command

Purpose

Removes **nroff**, **troff**, **tbl**, and **eqn** command constructs from files.

Syntax

deroff { **-ma -me -ms** [**-mm** [**-ml**]] } [**-i** | **-l**] [**-k**] [**-p**] [**-u**] [**-w**] [*File* ...]

Description

The **deroff** command reads the specified files (standard input by default) containing English-language text, removes all **troff** requests, macro calls, backslash constructs, **eqn** command constructs (between **.EQ** and **.EN** lines and between delimiters), and **tbl** command descriptions, then writes the remainder of the file to standard output.

The **deroff** command normally follows chains of included files (**.so** and **.nx troff** command requests). If a file has already been included, a **.so** request naming it is ignored and an **.nx** request naming that file ends execution.

> **Note:** The **deroff** command is not a complete **troff** command interpreter, so it can be confused by subtle constructs. Most errors result in too much rather than too little output.

Parameters

File Specifies English-language text files for the **deroff** command to remove the effects of **troff**, **eqn**, and **tbl** command processing. The default file is standard input.

Flags

-ma Ignores **MA** (**man**) macros in text so that only running text is output.
-me Ignores **ME** macros in text so that only running text is output. This is the default.
-ml Ignores **MM** macros in text (**-mm** flag) and also deletes **MM** list structures. The **-mm** flag must be specified with this flag.

Note: Do not use the **-ml** flag with nested lists.

-mm Ignores **MM** macros.
-ms Ignores **MS** macros in text so that only running text is output.
-i Suppresses the processing of included files.
-l Suppresses the processing of included files whose names begin with **/usr/lib**, such as macro files in **/usr/lib/tmac**.
-k Retains blocks specified to be kept together. The default is to remove kept blocks of text; for example, the **.ne** construct is removed.
-p Processes special paragraphs.
-u Removes the ASCII underline and boldface control sequences. This flag automatically sets the **-w** flag.
-w Makes the output a word list, with one word per line and all other characters deleted. Otherwise, the output follows the original.

In text, a word is any string that begins with a letter, contains at least two letters, and is composed of letters, digits, ampersands (&), and apostrophes ('). In a macro call, however, a word is a string that begins with at least two letters and contains a total of at least three letters. Delimiters are any characters other than letters, digits, punctuation, apostrophes, and ampersands. Trailing apostrophes and ampersands are removed from words.

Related Information

The **eqn** command, **neqn** command, **nroff** command, **tbl** command, **troff** command.

---

detachrset Command

Purpose

Detaches an rset from a process.

Syntax

detachrset [**-P**] *pid*

Description

The **detachrset** command detaches an rset from a process. Detaching an rset from a process will allow the process to use any of the processors and/or memory regions in the system.

Flags

-P Detaches the partition rset from the specified process (*pid*).

Parameters

pid Process ID.

Security

The user must have `root` authority or have CAP_NUMA_ATTACH capability and the target process must have the same effective **userid** as the command issuer. The user must have `root` authority to remove the partition rset from a process (the **-P** option).

Attention RBAC users and Trusted AIX users: This command can perform privileged operations. Only privileged users can run privileged operations. For more information about authorizations and privileges, see Privileged Command Database in *Security*. For a list of privileges and the authorizations associated with this command, see the **/etc/security/privcmds** database file.

Example

To detach the rset from process 21414, type:

```
detachrset 21414
```

Files

/usr/bin/detachrset	Contains the **detachrset** command.

Related Information

The **attachrset**, **execrset**, **lsrset**, **mkrset**, and **rmrset** commands.

---

devinstall Command

Purpose

Installs software support for devices.

Syntax

devinstall **-f** *File* **-d** *Device* [**-s**] [**-v**]

Description

The **devinstall** command installs software support for devices. It installs the software packages listed in the file specified by the **-f** flag.

For most new devices that are added after the initial software installation, the software for the new device can be installed using the **-i** flag of the **cfgmgr** command.

In some instances, the new device replaces a device that is needed to start the machine. For example, you might be replacing the SCSI adapter card that supports the root volume group or the graphics adapter card that supports the console. In this case, the machine will not start in normal mode until you have installed software support for this new device. To do this, turn your system off and install the new hardware according to the directions included with your hardware. Next, start up your machine in maintenance mode. During the startup process, the new adapter is detected and the **/tmp/device.pkgs** file is created containing the name of the software package needed to support the new hardware. Once the machine is in maintenance mode, you can install the software for this new device by running the **devinstall** command.

Flags

-f *File*	Specifies the file containing the list of packages to be installed. Typically, this will be the **/tmp/device.pkgs** file generated by the **cfgmgr** command.

-d *Device* Specifies where the installation medium can be found. This can be a hardware device, such as tape or diskette; it can be a directory that contains installation images; or it can be the installation image file itself. When the installation media is an IBM Installation tape or IBM Corrective Service tape, the tape device should be specified as no-rewind-on-close and no-retension-on-open. Examples of this would be **/dev/rmt0.1** for a high-density tape or **/dev/rmt0.5** for a low-density tape. For non-IBM-supplied tapes, use the options specified by the tape supplier.

-s Overwrites the **/var/adm/dev_pkg.fail** file. This file contains a list of all packages that did not install successfully and can be used to facilitate recovery or installation from a different source.

-v Specifies the verbose option, causing the **devinstall** command to display additional information while processing.

The **devinstall** command installs the device packages listed in the file specified on the command line. It runs the **geninstall** command with the **-I** ″acXge **/var/adm/ras/devinst.log**″, where a: apply, c: commit, X: extend fs, e: log and **/var/adm/ras/devinst.log** is the log file full path name, g: auto_include. (See the **geninstall** command for more information on these flags.) The **devinstall** command checks the summary file generated by the **geninstall** command for the results of each package install attempt and, based on this information, creates two files. The **/var/adm/dev_pkg.fail** file lists the packages that fail to install (if any). The **/var/adm/dev_pkg.success** file lists all packages that are installed successfully.

Return Values

A return value of 0 indicates that no packages were installed.

A return value of 1 indicates that at least one package was successfully installed, and the **bosboot** command should be executed.

A return value of 2 indicates that the **devinstall** command failed.

The **/var/adm/dev_pkg.success** file lists those packages that successfully installed. The **/var/adm/dev_pkg.fail** file lists those packages that failed installation.

Security

Privilege Control: Only the root user can run this command.

Examples

To install software to support a new device after you have started the machine from the device installation tape and entered maintenance mode, enter:

```
devinstall -f /../tmp/device.pkgs -d /dev/rmt0.1
```

Then, run the **bosboot** command.

```
bosboot -ad /dev/ipldevice
```

File

/dev/rmtn Specifies the raw streaming tape interface.

Related Information

The **bosboot** command, **cfgmgr** command, **installp** command.

devnm Command

Purpose
Names a device.

Syntax
devnm *Path* ...

Description
The **devnm** command reads the *Path* parameter, identifies the special file associated with the mounted file system where the *Path* parameter resides, and writes the special file name to standard output. Each *Path* parameter must be a full path name.

The most common use of the **devnm** command is by the **/etc/rc** command file to construct a mount table entry for the root device.

Note: This command is for local file systems only.

Examples
1. To identify the device on which a file resides, enter:
   ```
   devnm /diskette0/bob/textfile
   ```
   This displays the name of the special device file on which the `/diskette0/bob/textfile` file resides. If a diskette is mounted as the `/diskette0` device, the **devnm** command displays:
   ```
   fd0 /diskette0/bob/textfile
   rfd0 /diskette0/bob/textfile
   ```
   This means the `/diskette0/bob/textfile` file resides on the **/dev/fd0** diskette drive.
2. To identify the device on which a file system resides, enter:
   ```
   devnm /
   ```
   This displays the name of the device on which the root file system(/) resides. The following list is displayed on the screen:
   ```
   hd0 /
   ```
   This means that the root file system (/) resides on the **/dev/hd0** device.

Files

/dev	Specifies the directory.
/usr/sbin/devnm	Contains the **devnm** command.

Related Information
The **rc** command.

df Command

Purpose
Reports information about space on file systems. This document describes the AIX **df** command as well as the System V version of **df**.

Syntax

df [[**-P**] | [**-l** | **-M** | **-i** | **-t** | **-v**]] [**-k**] [**-m**] [**-g**] [**-s**] [*FileSystem* ... | *File...*]

Description

The **df** command displays information about total space and available space on a file system. The *FileSystem* parameter specifies the name of the device on which the file system resides, the directory on which the file system is mounted, or the relative path name of a file system. The *File* parameter specifies a file or a directory that is not a mount point. If the *File* parameter is specified, the **df** command displays information for the file system on which the file or directory resides. If you do not specify the *FileSystem* or *File* parameter, the **df** command displays information for all currently mounted file systems. File system statistics are displayed in units of 512-byte blocks by default.

The **df** command gets file system space statistics from the **statfs** system call. However, specifying the **-s** flag gets the statistics from the virtual file system (VFS) specific file system helper. If you do not specify arguments with the **-s** flag and the helper fails to get the statistics, the **statfs** system call statistics are used. Under certain exceptional conditions, such as when a file system is being modified while the **df** command is running, the statistics displayed by the **df** command might not be accurate.

Note: Some remote file systems, such as the Network File System (NFS), do not provide all the information that the **df** command needs. The **df** command prints blanks for statistics that the server does not provide.

The **df** command does not fully support NFSv4 filesystems. Use the **nfs4cl** command to extract block and space information.

Flags

-g Displays statistics in units of GB blocks. The output values for the file system statistics would be in floating point numbers as value of each unit in bytes is significantly high.

-i Displays the number of free and used i-nodes for the file system; this output is the default when the specified file system is mounted.

-l Displays information on the total number of blocks, the used space, the free space, the percentage of used space, and the mount point for the file system.

-k Displays statistics in units of 1024-byte blocks.

-m Displays statistics in units of MB blocks. The output values for the file system statistics would be in floating point numbers as value of each unit in bytes is significantly high.

-M Displays the mount point information for the file system in the second column.

-P Displays information on the file system in POSIX portable format.

When the **-P** flag is specified, the header line appears similar to:

```
Filesystem 512-blocks Used Available Capacity Mounted on\n
```

If the **-k**, **-m** or **-g** flag is specified in addition to the **-P** flag, the column heading 512-blocks is replaced by the respective units, depending on which of these flags is used with the **-P** flag.

File system statistics are displayed on one line in the following order:

FileSystem, TotalSpace, UsedSpace, FreeSpace, UsedPercentage, MountPoint

-s Displays statistics on unmounted JFS or Enhanced JFS file systems by the command line arguments. If there are no arguments specified, the **-s** flag has no effect. If the file systems specified by the argument are currently mounted or an argument is a file, the **-s** flag has no effect for that particular argument. To collect statistics on unmounted file systems, an argument must be a JFS or Enhanced JFS file system mount point or device, the file system must be listed in **/etc/filesystems**, and the user must have read access to the device.

-t Includes figures for total allocated space in the output.

-v Displays all information for the specified file system.

The values of the output parameters with the flags **-m** and **-g** would be rounded off to nearest second decimal digit. If all or any two of the **-k**, **-m** and **-g** flags are specified, the last one specified takes effect.

Exit Status

This command returns the following exit values:

0	Successful completion.
>0	An error occurred.

Examples

1. To display information about all mounted file systems, enter:

   ```
   df
   ```

   If your system has the **/**, **/usr**, **/site**, and **/usr/venus** file systems mounted, the output from the **df** command resembles the following:

   ```
   Filesystem 512-blocks Free   %Used   Iused  %Iused  Mounted on
   /dev/hd0    19368     9976    48%     4714    5%     /
   /dev/hd1    24212     4808    80%     5031   19%     /usr
   /dev/hd2     9744     9352     4%     1900    4%     /site
   /dev/hd3     3868     3856     0%      986    0%     /usr/venus
   ```

2. To display information about /test file system in 1024-byte blocks, enter:

   ```
   df -k /test
   Filesystem    1024 blocks    Free    %Used   Iused  %Iused  Mounted on
   /dev/lv11         16384     15824       4%      18      1%  /tmp/ravi1
   ```

   This displays the file system statistics in 1024-byte disk blocks.

3. To display information about /test file system in MB blocks, enter:

   ```
   df -m /test
   Filesystem    MB blocks    Free    %Used    Iused  %Iused  Mounted on
   /dev/lv11       16.00     15.46       4%       18      1%  /tmp/ravi1
   ```

   This displays file system statistics in MB disk blocks rounded off to nearest 2nd decimal digit.

4. To display information about the **/test** file system in GB blocks, enter:

   ```
   df -g /test
   Filesystem    GB blocks   Free     %Used    Iused  %Iused  Mounted on
   /dev/lv11          0.02   0.02        0%       18      1%  /tmp/ravi1
   ```

   This displays file system statistics in GB disk blocks rounded off to nearest 2nd decimal digit.

5. To display available space on the file system in which your current directory resides, enter:

   ```
   cd/
   df .
   ```

   The output from this command resembles the following:

   ```
   Device   512-blocks  free   %used   iused   %iused  Mounted on
   /dev/hd4    19368    9976    48%     4714    5%     /
   ```

Files

/etc/filesystems	Lists the known file systems and defines their characteristics.
/etc/vfs	Contains descriptions of virtual file system types.

Related Information

The **fsck** command.

The **filesystems** file.

The File systems in *Operating system and device management* explains file system types, management, structure, and maintenance.

The Mounting in *Operating system and device management* explains mounting files and directories, mount points, and automatic mounts.

System V df Command

Purpose

Reports number of free disk blocks and files.

Syntax

/usr/sysv/bin/df [**-a**] [**-l**] [[[**-e**] [**-g**] [**-n**]] | [[**-i**] [**-v**]] | **-t**]] [*FileSystem* ...] [*File* ...]

Description

The **df** command displays information about total space and available space on a file system. File system statistics are displayed in units of 512-byte blocks

Flags

-a	Performs the default operation and prints the mount point, the device name, number of free blocks and number of used inodes (files).
-e	Print only the number of free files.
-g	Print the entire **statvfs** structure. This option overrides the **-a** , **-e**, **-i**, **-n**, **-t** and **-v** options. The numbers for available, total, and free blocks are reported in 512 byte blocks.
-i	Displays the total number of inodes, the number of free inodes, the number of used inodes, and the percentage of inodes in use.
-l	Reports on local file systems only.
-n	Prints the type of filesystem.
-t	Causes total allocated block figures to be reported.
-v	Reports percent of blocks used as well as the number of blocks used and free.

Parameters

File	The *File* parameter specifies a file or a directory that is not a mount point. If the *File* parameter is specified, the **df** command displays information for the file system on which the file or directory resides.
FileSystem	The *FileSystem* parameter specifies the name of the device on which the file system resides, the directory on which the file system is mounted, or the relative path name of a file system.

Note: If the *FileSystem* or *File* parameter is not specified, the **df** command displays information for all currently mounted file systems.

Exit Status

0 The command completed successfully

>0 An error occurred.

Examples

1. To display information about all mounted file systems, enter:

   ```
   /usr/sysv/bin/df
   ```

   The output looks similar to the following:

   ```
   /              (/dev/hd4       ):     19656 blocks    1504 files
   /usr           (/dev/hd2       ):   1139904 blocks   20254 files
   /var           (/dev/hd9var    ):     23096 blocks     512 files
   /tmp           (/dev/hd3       ):      2464 blocks     204 files
   /home          (/dev/hd1       ):     44208 blocks     146 files
   /proc          (/proc          ):         0 blocks       0 files
   /opt           (/dev/hd10opt   ):     13880 blocks     310 files
   ```

2. To display information about the file system in which your current directory resides, enter:

   ```
   /usr/sysv/bin/df .
   ```

3. To display the total number of inode, the number of free inodes and the number of available inodes in all mounted file systems, enter:

   ```
   /usr/sysv/bin/df -i
   ```

   The output looks similar to the following:

   ```
   Mount Dir  Filesystem      iused    avail    itotal  %iused
   /          /dev/hd4         1504     6688      8192     19%
   /usr       /dev/hd2        20254   127202    147456     14%
   /var       /dev/hd9var       512     3584      4096     13%
   /tmp       /dev/hd3          204     5940      6144      4%
   /home      /dev/hd1          146    14190     14336      2%
   /proc      /proc               0        0         0       0
   /opt       /dev/hd10opt      310     5834      6144      6%
   ```

4. To display the total number of blocks , the number of used blocks and the number of free blocks on a the **/tmp** file system, enter:

   ```
   /usr/sysv/bin/df -v /tmp
   ```

5. To display the type of filesystem, enter:

   ```
   /usr/sysv/bin/df -n
   ```

6. To display inode information on all local filesystems, enter:

   ```
   /usr/sysv/bin/df -i -l
   ```

7. To display the statvfs structure information on all the filesystems, enter:

   ```
   /usr/sysv/bin/df -g
   ```

8. To display the number of free files on filesystems, enter:

   ```
   /usr/sysv/bin/df -e
   ```

Files

/usr/sysv/bin/df	Contains the System V **df** command.
/etc/filesystems	Contains filesystem information.

Related Information

The **/usr/bin/df** command.

dfmounts Command

Purpose

Displays mounted resource information.

Syntax

dfmounts [**-F** *fstype*] [**-h**] [*server* ...]

Description

The **dfmounts** command prints local systems that are remotely mounted by clients through Network File System (NFS). It also prints the list of clients that have mounted the resource.The **dfmounts** command prints a header that is followed by a list of resource information separated with whitespace characters within fields.

For each resource, the following fields are displayed:

RESOURCE
For NFS, a hyphen ″-″ is marked.

SERVER
Indicates the machine from which the resource was mounted.

PATHNAME
Indicates the path of the shared resource.

CLIENTS
A comma separated list of systems that currently have the resource mounted.

Flags

-F *fstype*	Specifies the File System Type (fstype). Only **nfs** type of filesystem is supported.
-h	Suppress the header line in the output of **dfmounts**.

Parameters

Server	Represents a system on the network that had made its resources available to the local system. *Server* prints the resources that is made available from the machine together with the current clients using each resource. If this parameter is not specified, then the **dfmounts** command prints information by assuming that server is the local system. Multiple server names can be provided with the **dfmounts** command.

Exit Status

0 The command completed successfully

>0 An error occurred.

Security

Examples

1. To print the mounted resource information on a the system ″mercury″ for filesytem type ″**nfs**″, enter:
   ```
   dfmounts -F nfs mercury
   ```
2. To print mounted resource information without header on the system for filesystem type ″**nfs**″, enter:

```
dfmounts -hF nfs
```

Files

/usr/bin/dfmounts	Contains the generic System V **dfmounts** command.
/usr/lib/fs/nfs/dfmounts	Contains the System V **dfmounts** command for nfs.
/etc/vfs	Contains the description for known virtual filesystem implementations.

Related Information

The **dfshares** command.

---

dfpd Command

Purpose

Provides load statistics about servers being load balanced to the Load Manager.

Syntax

/usr/sbin/dfpd [**-d**] [**-f** *ConfigurationFile*]

Description

The DFP daemon (**dfpd**) runs on the server being load balanced and provides load statistics about the server to the Load Manager. This enables the Load Manager to send future connections to the servers that are more available which helps in balancing the load.

When the **dfpd** daemon starts, it reads its configuration information from the file specified in the *ConfigurationFile* parameter. If the parameter is not specified, the **dfpd** daemon reads its configuration information from the **/etc/dfpd.conf** file.

Once started, the **dfpd** daemon listens for connections from the Load Manager on the port specified in the configuration file.

DFP daemon Configuration File

The **/etc/dfpd.conf** file can be updated by editing it. The entries in the **/etc/dfpd.conf** file include the following information:

The MD5 key entry specifies the secret key (up to 64 characters) that should be the same between the DFP clients, server and the Load Manager. An example of the MD5 key entry is:

```
md5key 1234567890abcdefabcdef12345678901234567890abcdefabcdef1234567890
```

The Load Manager listener entry specifies the port on which the DFP server listens for Load Manager connection. An example of the Load Manager entry is:

```
ldlistener 9503
```

The poll idle time entry specifies the period between successive computations of the CPU idle time. An example of the poll idle time entry is:

```
pollidletime 30
```

The computed idle time is multiplied by the *mfactor* value before reporting the time to the Load Manager. This is useful in rationalizing the weights among machines of different capacities. The default value is the number of CPUs on the host. An example of the mfactor entry is:

```
mfactor 1
```

Flags

-d	Runs in debug mode and does not become a daemon process.
-f *ConfigurationFile*	Causes the daemon to use the specified *ConfigurationFile*.

dfsck Command

Purpose

Checks and repairs two file systems simultaneously on different drives.

Syntax

dfsck [*FlagList1*] *FileSystem1* [*FlagList2*] *FileSystem2*

Description

The **dfsck** command lets you simultaneously check two file systems on two different drives. Use the *FlagList1* and *FlagList2* parameters to pass flags and parameters for the two sets of file systems. For a list of valid flags for *FlagList1* and *FlagList2*, see the flags section. Use a **-** (minus sign) to separate the file system groups if you specify flags as part of the arguments.

The **dfsck** command permits you to interact with two **fsck** commands at once. To aid in this, the **dfsck** command displays the file system name with each message. When responding to a question from the **dfsck** command, prefix your response with a `1` or a `2` to indicate whether the answer refers to the first or second file system group.

Attention: Do not use the **dfsck** command to check the root file system.

Flags

-d*BlockNumber*	Searches for references to a specified disk block. Whenever the **fsck** command encounters a file that contains a specified block, it displays the i-node number and all path names that refer to it.
-f	Performs a fast check. Under normal circumstances, the only file systems likely to be affected by halting the system without shutting down properly are those that are mounted when the system stops. The **-f** flag prompts the **fsck** command not to check file systems that were unmounted successfully. The **fsck** command determines this by inspecting the **s_fmod** flag in the file system superblock. This flag is set whenever a file system is mounted and cleared when it is unmounted successfully. If a file system is unmounted successfully, it is unlikely to have any problems. Because most file systems are unmounted successfully, not checking those file systems can reduce the checking time.
-i*i-NodeNumber*	Searches for references to a specified i-node. Whenever the **fsck** command encounters a directory reference to a specified i-node, it displays the full path name of the reference.
-n	Assumes a `no` response to all questions asked by the **fsck** command; does not open the specified file system for writing.

-o *Options*
Passes comma-separated options to the **fsck** command. These options are assumed to be file system implementation-specific, except that the following are currently supported for all file systems:

mountable
Causes the **fsck** command to exit with success, returning a value of 0, if the file system in question is mountable (clean). If the file system is not mountable, the **fsck** command exits returning with a value of 8.

mytype
Causes the **fsck** command to exit with success (0) if the file system in question is of the same type as either specified in the **/etc/filesystems** file or by the **-V** flag on the command line. Otherwise, 8 is returned. For example, `fsck -o mytype -V jfs /` exits with a value of 0 if **/** (the root file system) is a journaled file system.

-p
Does not display messages about minor problems but fixes them automatically. This flag does not grant the wholesale license that the **-y** flag does and is useful for performing automatic checks when the system is started normally. You should use this flag as part of the system startup procedures, whenever the system is being run automatically. Also allows parallel checks by group.

-t*File*
Specifies a *File* parameter as a scratch file on a file system other than the one being checked, if the **fsck** command cannot obtain enough memory to keep its tables. If you do not specify the **-t** flag and the **fsck** command needs a scratch file, it prompts you for the name of the scratch file. However, if you have specified the **-p** flag, the **fsck** command is unsuccessful. If the scratch file is not a special file, it is removed when the **fsck** command ends.

-V *VfsName*
Uses the description of the virtual file system specified by the *VFSName* variable for the file system instead of using the **/etc/filesystems** file to determine the description. If the **-V** *VfsName* flag is not specified on the command line, the **/etc/filesystems** file is checked and the **vfs=***Attribute* of the matching stanza is assumed to be the correct file system type.

-y
Assumes a yes response to all questions asked by the **fsck** command. This flag lets the **fsck** command take any action it considers necessary. Use this flag only on severely damaged file systems.

Security

Attention RBAC users and Trusted AIX users: This command can perform privileged operations. Only privileged users can run privileged operations. For more information about authorizations and privileges, see Privileged Command Database in *Security*. For a list of privileges and the authorizations associated with this command, see the **/etc/security/privcmds** database file.

Examples

1. To simultaneously check two file systems on two different drives, enter:

   ```
   dfsck  -p /dev/hd1 -  -p /dev/hd7
   ```

   This command checks both file systems simultaneously, if the file systems on the **/dev/hd1** and **/dev/hd7** devices are located on two different drives. You can also specify the file system names found in the **/etc/filesystems** file.

Files

/usr/sbin/dfsck	Contains the **dfsck** command.
/etc/filesystems	Lists the known file systems and defines their characteristics.
/etc/vfs	Contains descriptions of virtual file system types.
/etc/rc	Contains commands (including the **fsck** command) that are run when the system is started.

Related Information

The **fsck** command, **fsdb** command, **istat** command, **mkfs** command, **ncheck** command, **rc** command, **shutdown** command.

The **filesystems** file, **filsys.h** file.

The File systems in *Operating system and device management* explains file system types, management, structure, and maintenance.

For information on installing the Web-based System Manager, see Chapter 2: Installation and System Requirements in *AIX Version 6.1 Web-based System Manager Administration Guide*.

The System management interface tool in *Operating system and device management* explains the SMIT structure, main menus, and tasks.

---

dfshares Command

Purpose

Lists available resources from remote systems.

Syntax

dfshares [**-F** *FileSystemType*] [**-h**] [*Server* ...]

Description

The **dfshares** command provides information about resources that are available to the host through the Network File System. The **dfshares** command prints a header line, followed by a list of lines that contain white spaces as field separators.

For each resource, the following fields are displayed:

RESOURCE
Displays the resource name that is exported in the form of server:path.

SERVER
Displays the machine that is providing the resource.

ACCESS
Displays the access permissions granted to the client systems. However, **dfshares** cannot determine this information for a NFS resource and therefore populates the field with a hyphen (″-″).

TRANSPORT
Displays the transport provider over which the resource is shared. However, **dfshares** cannot determine this information for a NFS resource and therefore populates the field with a hyphen (″-″).

Flags

-F *FileSystemType*	Specifies the filesystem type. Only **nfs** type of filesystem is supported.
-h	Suppress the header line in the output of **dfshares**.

Parameters

Server — Represents a system on the network that has provided resources to the local machine. If this parameter is not specified, then the **dfshares** command prints the information for the local system, itself. More than one server name can be specified with **dfshares**.

Exit Status

0 The command completed successfully.

>0 An error occurred.

Examples

1. To print the resource information on the system ″mercury″ for an **nfs** type filesystem, enter:
   ```
   dfshares -F nfs mercury
   ```
2. To print resource information without header on the system, enter:
   ```
   dfshares -hF nfs
   ```

Files

/usr/bin/dfshares	Contains the generic System V **dfshares** command.
/usr/lib/fs/nfs/dfshares	Contains the System V **dfshares** command for filesystems of type **nfs**.
/etc/vfs	Contains the descriptions for known virtual filesystem implementations.

Related Information

The **dfmounts** command.

---

dhcpaction Command

Purpose

Provides a script that runs every time a client updates its lease.

Syntax

/usr/sbin/dhcpaction *HostName DomainName IPAddress LeaseTime ClientID* { **A** | **PTR** | **BOTH** | **NONE** } { **NONIM** | **NIM** }

Description

The **dhcpaction** command provides methods to update the DNS server by means of calling the **nsupdate** command with the proper sequence of events to update the `A` record, `PTR` record, or both. The **dhcpaction** command is called by the DHCP client and server daemons. It is called from the updateDNS string. This is configurable because in some environments, mainly heterogenous ones, some clients may not be able to update the `A` record or the `PTR` record. The default action is for the client to update the `A` record and the server to update the `PTR` record. The options may be set in the daemon configuration files to allow for any policy the network administator wants.

The **dhcpaction** command also allows you to run NIM and DHCP concurrently. The **dhcpaction** command, when given the NIM paramete, will try and issue updates to NIM objects when their IP addresses change. This keeps the objects in sync. To do this, some pending operations may have to be canceled. The objects will be commented and a message will be sent to the console of the master machine. The objects should not be reset often. Addresses should not commonly change in the DHCP environment. Only the clients should set the `NONIM` option.

Parameters

ClientID	Specifies the client ID to use when updating the DNS server.
DomainName	Specifies the domain name to use when updating the DNS server.
HostName	Specifies the host name to try and update in the DNS server.
IPAddress	Specifies the IP address to associate with the host name in the DNS server.
LeaseTime	Specifies the duration of the association between the host name and IP address in the DNS server in seconds.

Options

A | **PTR** | **BOTH** | **NONE**	Specifies which if any record should be updated in the DNS server.
NONIM | **NIM**	Specifies if the script should take actions to help NIM and DHCP interact correctly. This should only be set to `NIM` on DHCP Servers.

Exit Status

This command returns the following exit values:

0	Successful completion.
>0	An error occurred.

Security

Access Control: Any User, but may need to be root for some NIM actions

Files

/usr/sbin/dhcpaction	Contains the **dhcpaction** command.
/etc/dhcpcd.ini	Contains the DHCP Client Configuration File

Related Information

The **inetd** daemon, **dhcpsd** daemon, **dhcprd** daemon.

DHCP Client Configuration File

DHCP Server Configuration File

bootp Configuration File

TCP/IP address and parameter assignment - Dynamic Host Configuration Protocol

TCP/IP daemons in *Networks and communication management*.

---

dhcpcd Daemon

Purpose

Implements a Dynamic Host Configuration Protocol (DHCP) client. Serves addresses and configuration information to DHCP server.

Syntax

To Implement a DHCP Client by Using the System Resource Controller:

startsrc -s dhcpcd [**-a***Argument*] ...

To Implement a DHCP Client without Using the System Resource Controller:

dhcpcd [**-f** *ConfigurationFile*] [**-i** *IPAddress*] [**-l** *LeaseFile*] [**-n**] [**-o** *OptionsFile*] [**-r**] [**-t** *Seconds*] [**-T** *Minutes*]

Description

The **dhcpcd** daemon implements a DHCP client by setting up IP (Internet Protocol) addresses and other parameters by using the DHCP protocol.

The **dhcpcd** daemon is normally started by the **/etc/rc.net** file that normally runs at boot time. By default, this is commented out and not run on machine startup. There are smit options to enable the DHCP client.

The **dhcpcd** daemon reads its configuration file and attempts to bring up and get an IP address and other configuration options for the interfaces specified within the configuration file. The **dhcpcd** daemon runs in the background while the system is up. It will renew an already received address as required.

The **dhcpcd** daemon also runs in DHCP Inform mode when the **-i** flag is used. This mode lets a client retrieve configuration information from a DHCP server without getting an IP address. This is useful for static addresses, but not for dynamic items like print servers and other options. The **dhcpcd** daemon will run once for the specified address.

The **refresh** command can be used to cause the **dhcpcd** daemon to reread the configuration file. A SIGHUP may also be used to get the same response.

The default dhcpcd configuration file is **/etc/dhcpcd.ini**. It contains logging and network interface information.

You can use a Web-based System Manager application (**wsm network** fast path) to run this command. You could also use the System Management Interface Tool (SMIT) **smit usedhcp** fast path to run this command.

Flags

-f *ConfigurationFile*	Specifies the configuration file to be used. The default is the **/etc/dhcpcd.ini** file.
-i *IPAddress*	Specifies that the **dhcpcd** daemon should use DHCP Inform mode. The ip address tells DHCP which interface to get configuration information on.
-l *LeaseFile*	Specifies a different lease file. The lease file gets generated by the client when it obtains a lease. By default, the lease file is **/etc/dhcpc.db**.
-n	Prevents the interface from being reconfigured when it receives a new address.
-o *OptionsFile*	Specifies the options file. By default, the options file is **/etc/dhcpc.opt**.
-r	Brings the client daemon up then down when run one time.
-t *Seconds*	Specifies the amount of seconds that **dhcpcd** will wait before placing itself in the background. This allows a machine to continue booting if a DHCP Server cannot be found.
-T *Minutes*	Specifies the time in minutes. If the dhcp client fails to configure an address for an interface (for example, due to non-availability of dhcp server) within this timeout value, it stops further attempt.

Exit Status

This command returns the following exit values:

0 Successful completion.
>0 An error occurred.

Security

Access Control: You must have root authority to run this command.

Files

/usr/sbin/dhcpcd	Contains the **dhcpcd** daemon.
/etc/dhcpcd.ini	Contains the default client configuration file
/etc/services	Defines sockets and protocols used for internet services.
/etc/inetd.conf	Defines the services controlled by the **inetd** daemon.

Related Information

The **dhcpsconf** command.

The **startsrc** command, **stopsrc** command.

The **inetd** daemon, **dhcpsd** daemon, **dhcprd** daemon.

The /etc/inetd.conf file format, **/etc/services** file format.

DHCP Client Configuration File

DHCP Server Configuration File

bootp Configuration File

TCP/IP address and parameter assignment - Dynamic Host Configuration Protocol

The System Resource Controller in *Operating system and device management* gives an explanation of subsystems, subservers, and the System Resource Controller.

For information on installing the Web-based System Manager, see Chapter 2: Installation and System Requirements in *AIX Version 6.1 Web-based System Manager Administration Guide*.

TCP/IP daemons in *Networks and communication management*.

---

dhcpcd6 Daemon

Purpose

Implements a Dynamic Host Configuration Protocol for IPv6 (DHCPv6) client. Obtains IPv6 addresses and configuration information for an IPv6 node from DHCPv6 server.

Syntax

To Start a DHCPv6 Client by Using the System Resource Controller:

startsrc -s dhcpcd6 [**-a** *Argument*] ...

To Start a DHCPv6 Client without Using the System Resource Controller:

dhcpcd6 [**-f** *ConfigurationFileName*] [**-u** *Client_duid_File*] [**-p** *ClientPort*] [**-t** *SolicitTimeout*]

Description

The **dhcpcd6** daemon implements a DHCPv6 client by setting up IPv6 (Internet Protocol version 6) addresses and other parameters by using the DHCPv6 protocol.

The **dhcpcd6** daemon is normally started by the **/etc/rc.net** file that normally runs at boot time. By default, this is commented out and not run on machine startup. The **dhcpcd6** daemon runs in the background while the system is up.

The **dhcpcd6** daemon reads its configuration file and attempts to bring up and get one or more IPv6 addresses and other configuration options for the interfaces specified within the configuration file. The addresses obtained from the server are renewed as mandated by the server.

When a DHCPv6 client does not need to have a DHCPv6 server assign it IPv6 addresses, the client can obtain only configuration information such as a list of available DNS servers or NTP servers. This is useful when the node is configured with static addresses.

The **refresh** command can be used to cause the **dhcpcd6** daemon to reread the configuration file. A SIGHUP may also be used to get the same response.

The default **dhcpcd6** configuration file is **/etc/dhcpv6/dhcpc6.cnf**. It contains logging and network interface information.

Flags

-f *ConfigurationFileName*	Specifies the configuration file to be used. Default is **/etc/dhcpv6/dhcpc6.cnf**.
-p *ClientPort*	Specifies the client port to be used. Default is 546.
-t *SolicitTimeout*	Specifies the time until the client solicits configuration information from the server before exiting.
-u *Client_duid_File*	Specifies the client identifier file to be used. Default is **/etc/dhcpv6/dhcpc6.duid**.

Exit Status

This command returns the following exit values:

0	Successful completion.
>0	An error occurred.

Security

Access Control: You must have root authority to run this command.

Examples

1. To start the DHCPv6 client with the configuration file **dhcpcd6.cnf** located in **/usr/local**, type:

```
startsrc -s dhcpcd6 -a "-f /usr/local/dhcpcd6.cnf"
```

Location

/usr/sbin/dhcpcd6

Files

/usr/sbin/dhcpcd6	Contains the **dhcpv6** client daemon.
/etc/dhcpv6/dhcpc6.cnf	Contains the default configuration file.
/etc/dhcpv6/dhcpc6.db	Contains the client lease file. This file is created by the client daemon and is not configurable.
/etc/dhcpv6/dhcpc6.duid	Contains the client identifier file. This file is created by the client daemon and is not configurable.

Related Information

The “dhcpsdv6 Daemon” on page 106.

The **startsrc** command in *AIX Version 6.1 Commands Reference, Volume 5*.

---

dhcprd Daemon

Purpose

Forwards BOOTP and DHCP packets off the local network.

Syntax

To Forward Information to the DHCP Server by Using the System Resource Controller:

startsrc -s dhcprd [**-a** *Argument*] [**-a** *Argument*] ...

To Forward Information to the DHCP Server without Using the System Resource Controller:

dhcprd [**-f** *ConfigurationFile*]

Description

The **dhcprd** daemon listens for broadcast packets, receives them, and forwards them to the appropriate server. This keeps broadcasts from having to be propagated to other networks. The DHCP Relay Agent handles the forwarding the DHCP and BOOTP client broadcast packets off of the local network and on to a set of servers. The initial packets sent by a BOOTP or DHCP client are broadcasts on the local interface of the client machine. These packets are not allowed to be passed through network gateways and routers. A BOOTP/DHCP relay agent, the **dhcprd** daemon, sends these packets to the appropriate servers.

The DHCP Server reads **/etc/services** file to determine which port it should use for receiving requests. The default service is **dhcps**. Because this is the same port that the **bootpd** daemon uses, you can only have one (either **dhcprd** or **bootpd**) daemon running. If you choose the **dhcprd** daemon, you will need to uncomment **bootp** from the **/etc/inetd.conf** file, then type **refresh -s inetd** on the command line.

Note: If **bootpd** is running, this program needs to be stopped before starting the daemons.

Flags

-f *ConfigurationFile*	Specifies the configuration file to be used. The default is the **/etc/dhcprd.cnf** file.

Exit Status

This command returns the following exit values:

0 Successful completion.
>0 An error occurred.

Security

Access Control: You must have root authority to run this command.

Files

/usr/sbin/dhcprd	Contains the **dhcprd** daemon.
/etc/dhcprd.cnf	Contains the default configuration file.
/etc/services	Defines sockets and protocols used for internet services.
/etc/inetd.conf	Defines the services controlled by the **inetd** daemon.

Related Information

The **dhcpsconf** command, **startsrc** command, **stopsrc** command.

The **dhcpcd** daemon, **dhcpsd** daemon, **inetd** daemon.

DHCP Client Configuration File

DHCP Server Configuration File

TCP/IP address and parameter assignment - Dynamic Host Configuration Protocol

The System Resource Controller in *Operating system and device management* gives an explanation of subsystems, subservers, and the System Resource Controller.

For information on installing the Web-based System Manager, see Chapter 2: Installation and System Requirements in *AIX Version 6.1 Web-based System Manager Administration Guide*.

TCP/IP daemons in *Networks and communication management*.

---

dhcpsconf Command

Purpose

Simplifies DHCP (Dynamic Host Configuration Protocol) server configuration through a Graphical User Interface.

Syntax

dhcpsconf

Description

The **dhcpsconf** command brings up an X-windows GUI (Graphical User Interface) that lets the network administrator read, save, and modify configuration files. It also lets you start, stop, and retrieve statistics from a running server.

The **dhcpsconf** command displays a set of lists. The lists on the left show the available options and keys. The **dhcpsconf** command reads the **/etc/options.file** to determine its basic options and keys and starts with these as generic resource types. The GUI lets the network administrator define a set of named resources by selecting the resource menu button.

The resource definition dialog box lets the network administrator generate all the options and specifics that are on the networks. The network administrator can define and name the network, printers, name servers, dhcp servers, and other valid resource objects. Once this is done, these new resources are added to the key and option display on the main panel. These can be used to generate a server configuration file or set of server configuration files.

The GUI starts with an empty master file. A master file may contain either a single server or the definition of many servers and one actual server readable file. The master file is readable by one DHCP server, but multiple server information can be stored in it. This lets the network administator configure a single server image of the network, create a set of servers to handle the same set of data, and view and maintain it all in one file.

Options and keys are added to the server window by selecting the key or option, selecting where in the edit window the option or key should go, and selecting the add button corresponding to the key or option section. The option is added to the edit window at the position specified. If the item is a named resource, then it is added as is. If the item is one of the standard defaults, then a window requesting a value for the item appears.

DHCP servers are added just like other keys, except that they specify machines in the network that will be responsible for the items within their scope. The keys have scoping and syntactic ordering. Comments are not really keys, but they are allowed anywhere.

A server may have a network, class, client, or options specified within it. A network may have a subnet, class, client, or option. A subnet may have a class, client, or options. A class and client may only have options.

The servers have a set of configuration parameters that only apply to them. These are specified by the DHCP server key in the key list, or by using the default server options under the Server menu bar. The default server options apply to the master file. A DHCP Server specified within the master file receives the default options, but may be modified.

Any item placed in the Edit window may be edited, renamed, viewed, or deleted. This lets you place an item, see if it looks appropriate and make changes as necessary.

Upon completion of the configuration file, a single master file may be saved and/or a set of server files may be generated. The File menu button and server menu button both have save options. The File save button is for saving the master file. The Server save button is for saving a particular server to a file.

The File menu button also contains a quit option, an open option to retrieve a file, and a new option to erase everything created so far.

The Operations menu button contains a status button, a start button, a stop button, a refresh, and a send configuration file button. From these buttons, a remote server can report status, refresh itself with a new configuration file, may be stopped, and a configuration sent and restarted.

The Help button contains a set of help statements describing each of the windows items.

Exit Status

This command returns the following exit values:

0 Successful completion.
>0 An error occurred.

Security

Access Control: Any User

Files

/usr/sbin/dhcpsconf	Contains the **dhcpsconf** command.
/etc/dhcpcd.cnf	Contains the default client configuration file

Related Information

The **dhcpcd** daemon, **dhcprd** daemon, **dhcpsd** daemon, and **inetd** daemon.

DHCP Client Configuration File

DHCP Server Configuration File

TCP/IP address and parameter assignment - Dynamic Host Configuration Protocol

---

dhcpsd Daemon

Purpose

Implements a Dynamic Host Configuration Protocol (DHCP) server. Serves addresses and configuration information to DHCP clients.

Syntax

To Serve Information to the DHCP Clients by Using the System Resource Controller:

startsrc -s dhcpsd [**-a** *Argument*] [**-a** *Argument*] ...

To Serve Information to the DHCP Clients without Using the System Resource Controller:

dhcpsd [**-f** *ConfigurationFile*]

Description

The DHCP Server handles the assignment and maintenance of dynamic address assignment. It also handles the distribution of additional configuration information. The **dhcpsd** daemon runs in the background and maintains a database of server information that contains logging parameters, IP(Internet Protocol) address ranges, other network configuration information, and accessability information. The initial database is specified by the configuration file. The configuration file contains all the data to start configuring DHCP clients.

The DHCP Server maintains a database of addresses it has given out as well as who has them. These databases are kept in the files **/etc/dhcpsd.ar** and **/etc/dhcpsd.cr**. A server on startup will read the configuration file and setup its initial database of available addresses. The server accepts the **refresh** command or a SIGHUP signal to reread the configuration file.

The DHCP Server reads **/etc/services** file to determine which port it should use for receiving requests. The default service is dhcps. Because this is the same port that the **bootpd** daemon uses, you can only have one (either **dhcpsd** or **bootpd**) daemon running. If you choose the **dhcpsd** daemon, you will need to comment **bootp** from the **/etc/inetd.conf** file, then enter **refresh -s inetd** on the command line.

Note: If **bootpd** is running, this program needs to be stopped before starting the daemons.

Flags

-f *ConfigurationFile* Specifies the configuration file to be used.

Exit Status

This command returns the following exit values:

0 Successful completion.
>0 An error occurred.

Security

Access Control: You must have root authority to run this command.

Files

/usr/sbin/dhcpsd Contains the **dhcpsd** daemon.
/etc/services Defines sockets and protocols used for internet services.
/etc/inetd.conf Defines the services controlled by the **inetd** daemon.

Related Information

The **dhcpsconf** command

The **startsrc** command, **stopsrc** command.

The **dhcpcd** daemon, **dhcprd** daemon, **inetd** daemon.

DHCP Client Configuration File

DHCP Server Configuration File

TCP/IP address and parameter assignment - Dynamic Host Configuration Protocol

The System Resource Controller in *Operating system and device management* gives an explanation of subsystems, subservers, and the System Resource Controller.

TCP/IP daemons in *Networks and communication management*.

---

dhcpsdv6 Daemon

Purpose

Implements a Dynamic Host Configuration Protocol (DHCPv6) server. Serves addresses and configuration information to DHCPv6 clients.

Syntax

To Serve Information to the DHCPv6 Clients by Using the System Resource Controller:

startsrc -s dhcpsdv6 [**-a** *"Argument"*]

To Serve Information to the DHCP Clients without Using the System Resource Controller:

dhcpsdv6 [**-d**] [**-f** *ConfigurationFile*] [**-a** *DadminPort*] [**-p** *ServerPort*]

Description

The DHCPv6 Server handles the assignment and maintenance of dynamic address assignment. It also handles the distribution of additional configuration information. The **dhcpsd** daemon runs in the background and maintains a database of server information that contains logging parameters, IP (Internet Protocol) address ranges, other network configuration information, and accessibility information. The initial database is specified by the configuration file. The configuration file contains all the data to start configuring DHCP clients.

The DHCPv6 Server maintains a database of addresses it has given out as well as who has them. These databases are kept in the files **/etc/dhcpv6/db_file.crbk** and **/etc/dhcpv6/db_file.cr**. A server on startup will read the configuration file and setup its initial database of available addresses. The server accepts the refresh command or a SIGHUP signal to reread the configuration file.

Flags

-a	Specifies the Dadmin port; by default it is 942.
-d	Displays debugging information.
-f *ConfigurationFile*	Specifies the configuration file to be used. By default, the configuration file is **/etc/dhcpv6/dhcpsdv6.cnf**.
-p	Specifies the port used by the server to listen for incoming request; by default it is 547.

Exit Status

This command returns the following exit values:

0	Successful completion.
>0	An error occurred.

Security

Access Control: You must have root authority to run this command.

Examples

1. To start the DHCPv6 server with the configuration file **dhcpsdv6.cnf** located in **/usr/local**, type:

```
startsrc -s dhcpsdv6 -a "-f /usr/local/dhcpsdv6.cnf"
```

Location

/usr/sbin/dhcpsdv6

Files

/usr/sbin/dhcpsdv6	Contains the **dhcpsdv6** daemon.

/etc/dhcpv6/db_file.cr	Contains the client records. This file is created by the server daemon and is *not* configurable.
/etc/dhcpv6/db_file.crbk	Contains the client records. This file is created by the server daemon and is *not* configurable.
/etc/dhcpv6/dhcpsdv6.duid	Contains the server identifier file. This file is created by the server daemon and is *not* configurable.

Related Information

The “dhcpcd6 Daemon” on page 100.

The **startsrc** command in *AIX Version 6.1 Commands Reference, Volume 5*.

---

diag Command

Purpose

Performs hardware problem determination.

Syntax

diag [[**-a**] | [**-s** [**-c**]] [**-E** *days*] [**-e**] | [**-d** *Device* [**-c**] [**-v**] [**-e**] [**-A**]] | [**-B** [**-c**]] | [**-T** *taskname*] | [**-S** *testsuite*] | [**-c -d** *Device* **-L pending | complete**]

Description

The **diag** command is the starting point to run a wide choice of tasks and service aids. Most of the tasks/service aids are platform specific. The following tasks and service aids are available:

- Run Diagnostics
- Display or Change Diagnostic Run Time Options
- Display Service Hints
- Display Previous Diagnostic Results
- Display Hardware Error Report
- Display Software Product Data
- Display Configuration and Resource List
- Display Hardware Vital Product Data
- Display Resource Attributes
- Change Hardware Vital Product Data
- Format Media
- Certify Media
- Display Test Patterns
- Local Area Network Analyzer
- Add Resource to Resource List
- Delete Resource from Resource List
- SCSI Bus Analyzer
- Download Microcode
- Display or Change Bootlist
- Periodic Diagnostics
- Backup and Restore Media
- Disk Maintenance
- Configure Dials and LPFkeys

- Add or Delete Drawer Config
- Create Customized Configuration Diskette
- Update Disk Based Diagnostics
- Configure ISA Adapter
- Operating System Shell Prompt (Online Service Mode only)
- Display or Change Multiprocessor Configuration
  - Enable and disable individual processors
- Display or change BUMP Configuration
  - Update the flash EPROM with a new binary image
  - Display or change diagnostic modes
  - Display or change remote phone numbers and modem configurations
- Display or Change Electronic Mode Switch
- Process Supplemental Media (Standalone Mode only)
- Generic Microcode Download
- Run Error Log Analysis
- Service Aids for Use with Ethernet
- Update System Flash in AIX 5.1 and earlier (RSPC)
- Configure Ring Indicate Power-On in AIX 5.1 and earlier (RSPC)
- Configure Service Processor in AIX 5.1 and earlier (RSPC)
- Save or Restore Service Processor Configuration in AIX 5.1 and earlier (RSPC)
- Display Machine Check Error Log in AIX 5.1 and earlier (RSPC)
- 7135 RAIDiant Array Service Aids
- SCSI Device Identification and Removal
- SCSD Tape Drive Service Aid
- Escon Bit Error Rate Service Aid
- SSA Service Aid
- PCI RAID Physical Disk Identify
- Configure Ring Indicate Power On Policy (CHRP)
- Configure Surveillance Policy (CHRP)
- Configure Reboot Policy (CHRP)
- Configure Remote Maintenance Policy (CHRP)
- Save or Restore Hardware Management Policies (CHRP)
- Display Firmware Device Node Information (CHRP)
- Spare Sector Availability
- Update System or Service Processor Flash (CHRP)
- Display System Environmental Sensors (CHRP)
- Display Checkstop Analysis Results
- Analyze Adapter Internal Log
- Log Repair Action
- Flash SK-NET FDDI Firmware
- Display Microcode Level

You can use the Devices application in Web-based System Manager (wsm) to change device characteristics. You could also use the System Management Interface Tool (SMIT) **smit diag** fast path to run this command.

Flags

Note: Most users do not need to use any flags since the **diag** command is a menu driven program.

-A Specifies Advanced mode. Must also specify a device using the **-d** flag.

-a Processes any changes in the hardware configuration by asking if missing resources have been removed, turned off, and so on. In AIX 5.2 and higher, missing resources (indicated by an 'M') and missing resource paths (indicated by a 'P') are integrated into the diagnostic resource selection list.

-B Instructs diagnostics to run the base system test. Error log analysis are also done on areas in the base system that supports error log analysis.

-c Indicates that the machine will not be attended. No questions will be asked. Results are written to standard output. Must also use an optional flag that specifies a device to be tested, (**d**, **B**, **s**).

-d *Device* Specifies the device to run diagnostics on.

-E *Days* Specifies the number of days to use when searching the error log during Run Error Log Analysis. This flag works with any other flag.

-e Performs error log analysis if supported on the selected device. No tests are performed. Must be used with the **-d** flag, otherwise the resource selection menu displays. If used with the **-v** flag, **-v** takes precedence and the **-e** flag is ignored. .

-S *testsuite* Indicates a particular Test Suite of devices to test:

1. Base System
2. I/O Devices
3. Async Devices
4. Graphic Devices
5. SCSI Devices
6. Storage Devices
7. Commo Devices
8. Multimedia Devices

-L pending | complete Log Repair Action for a resource specified with the **-d** and **-c** options. Use **pending** if the part has been replaced, but it is not yet known if this part will remain in the system. Use **complete** if the part has been replaced and it is known that this part will remain in the system.

-s Runs diagnostics on all resources.

-T *taskname* Fastpath to specific task to run. Current fastpath tasks are the following:

format Format Media Task

certify Certify Media Task

download
Download Microcode Task

disp_mcode
Display Microcode Level Task

chkspares
Spare Sector Availability Task

identifyRemove
Hot Plug Task

Note: Tasks are platform and device dependent. Some tasks may not be available on the system.

-v Runs diagnostics in System Verification Mode, no error log analysis performed. The default is Problem Determination mode that tests the device and runs error log analysis. If used with the **-e** flag, the **-v** flag takes precedence and the **-e** flag is ignored. Must be used with the **-d** flag to specify a device to run diagnostics on.

Security

Access Control: Only the root user can run this command.

Privilege Control: System group.

Examples

To run diagnostics on the `scdisk0` device, without questions, enter:

```
diag -d scdisk0 -c
```

File

/usr/sbin/diag	Contains the **diag** command.

Related Information

The **diaggetrto** command, **diagsetrto** command.

For information on installing the Web-based System Manager, see Chapter 2: Installation and System Requirements in *AIX Version 6.1 Web-based System Manager Administration Guide*.

---

diaggetrto Command

Purpose

Displays diagnostic run-time options.

Syntax

diaggetrto [[**-a**] [**-d**] [**-l**] [**-m**] [**-n**] [**-p**] [**-s**]]

Description

The **diaggetrto** command displays the value of one or more diagnostic run time options. The following run-time options can be displayed with the **diaggetrto** command:

Display Diagnostic Mode Selection Menus
When this option is off, diagnostics run in Problem Determination mode only. The default is on.

Include Advanced Diagnostics
When this option is on, diagnostics run in advanced mode when run from the Task Selection Menu or command line. The default is off.

Number of days used to search error log
This option controls how old error log entries must be before they are no longer analyzed by diagnostics. The default is 7.

Display Progress Indicators
When this option is on, diagnostic applications that support progress indicators will display them. The default is on.

Diagnostic Event Logging
When this option is on, diagnostics log events. The default is on.

Diagnostic Event Log file size
This option controls the maximum size of the diagnostic event log. Allowable sizes are in increments of hundreds of kilobytes. The default is 100K.

Flags

-a	Displays the value of **Include Advanced Diagnostics**.
-d	Displays the value of **Diagnostic Event Logging**.
-l	Displays the value of **Diagnostic Event Log file size**.
-m	Displays the value of **Display Diagnostic Mode Selection Menus**.
-n	Displays the value of **Number of days used to search error log**.
-p	Displays the value of **Display Progress Indicators**.
-s	Displays all of the diagnostic run-time options.

Exit Status

0 The command completed successfully.

>0 An error occurred.

Examples

1. To display the diagnostic event log size, type:
   ```
   /usr/lpp/diagnostics/bin/diaggetrto -l
   ```
2. To check if progress indicators are turned on and to check if diagnostic event logging is turned on, type:
   ```
   /usr/lpp/diagnostics/bin/diaggetrto -p -d
   ```
3. To display the number of days to search the error log, type:
   ```
   /usr/lpp/diagnostics/bin/diaggetrto -n
   ```

Files

/usr/lpp/diagnostics/bin/diaggetrto	Contains the **diagsetrto** command.

Related Information

The **diagsetrto** command, **diag** command.

---

diagrpt Command

Purpose

Displays previous diagnostic results.

Syntax

diagrpt [[**-o**] | [**-s** *mmddyy*] | [**-a**] | [**-r**]]

Description

The **diagrpt** command displays the results of previous diagnostic sessions. There are three types of results that can be viewed:

- Diagnostic result files stored in **/etc/lpp/diagnostic/data** directory.
- Diagnostic Event Log Information.
- Diagnostic results stored in NVRAM on CHRP systems.

Flags

-o	Displays the last diagnostic results file stored in the **/etc/lpp/diagnostics/data** directory.
-s *mmddyy*	Displays all diagnostic result files logged since the date specified.
-a	Displays the long version of the Diagnostic Event Log.
-r	Displays the short version of the Diagnostic Event Log.

Examples

1. To list all previous diagnostic result files since Jan 31, 1999, enter:
   ```
   /usr/lpp/diagnostics/bin/diagrpt -s 013199
   ```
2. To view the short version of the diagnostic event log, enter:
   ```
   /usr/lpp/diagnostics/bin/diagrpt -r
   ```

File

/usr/lpp/diagnostics/bin/diagrpt	Contains the **diagrpt** command.

Related Information

The **diag** command.

---

diagsetrto Command

Purpose

Sets diagnostic run-time options.

Syntax

diagsetrto [[**-a on** | **off**] [**-d on** | **off**] [**-l** *Size*] [**-m on** | **off**] [**-n** *Days*] [**-p on** | **off**]]

Description

The **diagsetrto** command sets the value of any number of diagnostic run-time options. The following run-time options can be altered with the **diagsetrto** command:

Display Diagnostic Mode Selection Menus
When this option is off, diagnostics run in Problem Determination mode only. The default is on.

Include Advanced Diagnostics
When this option is on, diagnostics run in advanced mode when run from the Task Selection Menu or command line. The default is off.

Number of Days Used to Search Error Log
This option controls how old error log entries must be before they are no longer analyzed by diagnostics. The default is 7.

Display Progress Indicators
When this option is on, diagnostic applications that support progress indicators will display them. The default is on.

Diagnostic Event Logging
When this option is on, diagnostics log events. The default is on.

Diagnostic Event Log File Size
This option controls the maximum size of the diagnostic event log. Allowable sizes are in increments of hundreds of kilobytes. The default is 100K.

Flags

-a on | **off**	Sets the value of **Include Advanced Diagnostics**.
-d on | **off**	Sets the value of **Diagnostic Event Logging**.
-l *Size*	Sets the value of **Diagnostic Event Log file size**.
-m on | **off**	Sets the value of **Display Diagnostic Mode Selection Menus**.
-n *Days*	Sets the value of **Number of days used to search the error log**.
-p on | **off**	Sets the value of **Display Progress Indicators**.

Exit Status

0	The command completed successfully.
>0	An error occurred.

Examples

1. To set the diagnostic event log size to `500K`, type:
   ```
   /usr/lpp/diagnostics/bin/diagsetrto -l 500
   ```
2. To turn off progress indicators and turn off diagnostic event logging, type:
   ```
   /usr/lpp/diagnostics/bin/diagsetrto -p off -d off
   ```
3. To set the number of days to search the error log to `50`, type:
   ```
   /usr/lpp/diagnostics/bin/diagsetrto -n 50
   ```

Files

/usr/lpp/diagnostics/bin/diagsetrto	Contains the **diagsetrto** command.

Related Information

The **diaggetrto** command, **diag** command.

---

diction Command

Purpose

Highlights unclear or wordy sentences.

Syntax

diction [**-ml**] [**-mm**] [**-f** *PatternFile*] [**-n**] *File* ...

Description

The **diction** command finds all sentences in an English-language document that contain phrases from a database of unclear or wordy diction. Each phrase is bracketed with [ ] (brackets). Because the **diction** command runs the **deroff** command before looking at the text, include header files that contain appropriate formatting information as part of the input. The **explain** command provides an interactive thesaurus for the phrases found by the **diction** command.

Use of nonstandard formatting macros may cause incorrect sentence breaks. In particular, the **diction** command does not understand the **-me** flag.

Flags

-f *PatternFile*	Specifies a file containing examples of unclear diction; this file is used in addition to the default file.
-ml	Causes the **deroff** command to skip **mm** macro lists; can be used if a document contains many lists of sentence fragments.
-mm	Overrides the default **ms** macro package.
-n	Suppresses the use of the default file when used with the **-f** flag; only the file specified by the *PatternFile* parameter is used.

Files

/usr/lib/dict.d	Contains default pattern.

Related Information

The **deroff** command, **explain** command.

The **ms** macro package.

---

diff Command

Purpose

Compares text files.

Syntax

To Compare the Contents of Two Files

diff [**-c**| **-C** *Lines* | **-D** [*String*] | **-e** | **-f** | **-n**] [**-b**] [**-i**] [**-t**] [**-w**] *File1 File2*

diff [**-h**] [**-b**] *File1 File2*

To Sort the Contents of Directories and Compare Files That Are Different

diff [**-c** | **-C** *Lines* | **-e** | **-f** | **-n**] [**-b**] [**-i**] [**-l**] [**-r**] [**-s**] [**-S** *File*] [**-t**] [**-w**] *Directory1 Directory2*

diff [**-h**] [**-b**] *Directory1 Directory2*

Description

The **diff** command compares text files. It can compare single files or the contents of directories.

Note: The **diff** command only works with input files that are text files.

If the *Directory1* and *Directory2* parameters are specified, the **diff** command compares the text files that have the same name in both directories. Binary files that differ, common subdirectories, and files that appear in only one directory are listed.

When the **diff** command is run on regular files, and when comparing text files that differ during directory comparison, the **diff** command tells what lines must be changed in the files to make them agree. If neither the *File1* nor *File2* parameter is a directory, then either may be given as - (minus sign), in which case the standard input is used. If the *File1* parameter is a directory, then a file in that directory whose file name is the same as the *File2* parameter is used.

The typical output contains lines of these forms:

Lines Affected in File1	**Action**	**Lines Affected in** File2
Number1	**a**	Number2[**,**Number3]
Number1[**,**Number2]	**d**	Number3
Number1[**,**Number2]	**c**	Number3[**,**Number4]

These lines resemble **ed** subcommands to convert *File1* into *File2*. The numbers before the action letters pertain to *File1*; those after pertain to *File2*. Thus, by exchanging **a** for **d** and reading from right to left, you can also tell how to convert *File2* into *File1*. As in the **ed** command, identical pairs (where *Number1* = *Number2*) are abbreviated as a single number.

Following each of these lines, the **diff** command displays all lines affected in the first file preceded by a **<:** (less than sign, colon), then displays all lines affected in the second file are preceded by a **>** (greater than sign).

An exit value of 0 indicates no differences, 1 indicates differences found, and 2 indicates an error.

Note: If more than one of the **-c**, **-C**, **-D**, **-e**, **-f**, or **-n** flags are specified, the last one on the command line takes precedence. The system does not issue an error message.

Flags

-b Causes any amount of white space at the end of a line to be treated as a single newline character (the white-space characters preceding the newline character are ignored) and other strings of white-space characters, not including newline characters, to compare equally.

-C *Lines* Produces a **diff** command comparison with a number of lines of context equal to the value specified by the *Lines* variable. The **-C** flag modifies the output slightly. The output begins with identification of the files involved and their creation dates. Each change is separated by a line with a dozen * (asterisks). The lines removed from *File1* are marked with a - (minus sign) and those added to *File2* are marked with a + (plus sign). Lines changed from one file to the other are marked in both files with an ! (exclamation point). Changes that lie within the specified context lines of each other are grouped together as output.

-c Produces a **diff** command comparison with three lines of context. The **-c** flag modifies the output slightly. The output begins with identification of the files involved and their creation dates. Each change is separated by a line with a dozen * (asterisks). The lines removed from *File1* are marked with a - (minus sign) and those added to *File2* are marked with a + (plus sign). Lines changed from one file to the other are marked in both files with an ! (exclamation point). Changes within the specified context lines of each other are grouped together as output.

-D [*String*] Causes the **diff** command to create a merged version of *File1* and *File2* on the standard output. The C preprocessor controls are included so that a compilation of the result without defining *String* is equivalent to compiling *File1*, while defining *String* yields *File2.*

-e Produces output in a form suitable for use with the **ed** editor to convert *File1* to *File2*. When using this flag, the following shell program may help maintain multiple versions of a file. Only an ancestral file (**$1**) and a chain of version-to-version **ed** scripts (**$2**, **$3**, ...) made by the **diff** command need to be on hand. The latest version appears on the standard output as follows:

```
(shift; cat $*; echo '1,$p') | ed - $1
```

Extra commands are added to the output when the **-e** flag is used to compare directories, so the result is a shell script for converting text files that are common to the two directories from their state in *Directory1* to their state in *Directory2.*
Note: Editing scripts produced by the **-e** or **-f** flags cannot create lines consisting of a single . (period).

-f Produces output in a form not suitable for use with the **ed** editor, showing the modifications necessary to convert *File1* to *File2* in the reverse order of that produced under the **-e** flag.

-h	Performs an alternate comparison that may be faster if the changed sections are short and well separated. The **-h** flag works on files of any length. The **-c**, **-C**, **-D**, **-e**, **-f**, and **-n** flags cannot be used with the **-h** flag. All other flags except the **-b** flag are ignored when used with the **-h** flag.
-i	Ignores the case of letters. For example, a lowercase **a** is treated the same as an uppercase **A**.
-l	Long output format. Each result from the **diff** command text file comparison is piped through the **pr** command for pagination. Other differences are remembered and summarized after all text file differences are reported.
-n	Produces output similar to that of the **-e** flag, but in the opposite order and with a count of changed lines on each insert or delete command. This is the form used by the revision control system (RCS).
-r	Causes application of the **diff** command recursively to common subdirectories encountered.
-s	Reports files that are the same and otherwise not mentioned.
-S [*File*]	Ignores files whose names collate before the file specified by the *File* variable when comparing directories. The **-S** flag only applies to the directories specified in the *Directory1* and *Directory2* parameters. If you use the **-r** flag with the **-S** flag, the **-S** flag does not work recursively in the *Directory1* and *Directory2* subdirectories.
-t	Expands tabs in output lines. Typical output or the **-c** flag output adds characters to the front of each line, which may affect indentation of the original source lines and makes the output listing difficult to interpret. This flag preserves the original source's indentation.
-w	Ignores all spaces and tab characters and treats all other strings of blanks as equivalent. For example, `if ( a == b )` compares equally to `if(a==b)`.

Exit Status

This command returns the following exit values:

0	No differences were found.
1	Differences were found.
>1	An error occurred.

Examples

1. To compare two files, enter:

   ```
   diff chap1.back chap1
   ```

   This displays the differences between the files `chap1.bak` and `chap1`.
2. To compare two files while ignoring differences in the amount of white space, enter:

   ```
   diff  -w prog.c.bak prog.c
   ```

   If two lines differ only in the number of spaces and tabs between words, the **diff -w** command considers them to be the same.
3. To create a file containing commands that the **ed** command can use to reconstruct one file from another, enter:

   ```
   diff  -e chap2 chap2.old >new.to.old.ed
   ```

   This creates a file named `new.to.old.ed` that contains the **ed** subcommands to change `chap2` back into the version of the text found in `chap2.old`. In most cases, `new.to.old.ed` is a much smaller file than `chap2.old`. You can save disk space by deleting `chap2.old`, and you can reconstruct it at any time by entering:

   ```
   (cat new.to.old.ed ; echo '1,$p') | ed - chap2 >chap2.old
   ```

The commands in parentheses add `1,$p` to the end of the editing commands sent to the **ed** editor. The `1,$p` causes the **ed** command to write the file to standard output after editing it. This modified command sequence is then piped to the **ed** command (`| ed`), and the editor reads it as standard input. The **-** flag causes the **ed** command not to display the file size and other extra information because it would be mixed with the text of `chap2.old`.

Files

/usr/bin/diff	Contains the **diff** command.

Related Information

The **bdiff** command, **cmp** command**, diff3** command, **ed** command**, pr** command.

Files in *Operating system and device management* introduces you to files and the way you can work with them.

Input and output redirection in *Operating system and device management* describes how the operating system processes input and output.

---

diff3 Command

Purpose

Compares three files.

Syntax

diff3 [**-e** | **-x** | **-E** | **-X** | **-3**] *File1 File2 File3*

Description

The **diff3** command compares three files and writes to standard output the ranges of text that differ, flagged with the following codes:

====	All three files differ.
====1	*File1* differs.
====2	*File2* differs.
====3	*File3* differs.

The type of change needed to convert a given range of a given file to match another file is indicated in one of these two ways in the output:

File:*Number1* **a**	Text is to be added after line number *Number1* in *File*, where *File* is **1**, **2**, or **3**.
File:*Number1*[,*Number2*]**c**	Text in the range line *Number1* to line *Number2* is to be changed. If *Number1* is the same as *Number2*, the range may be abbreviated to *Number1*.

The contents of the range follows a **c** indication. When the contents of two files are identical, the **diff3** command does not show the contents of the lower-numbered file, although it shows the location of the identical lines for each.

Note: Edit scripts produced by the **-e** flag cannot create lines consisting of a . (period).

Flags

-3 Produces an edit script to incorporate only changes flagged ====3.

-E, **-X** These are similar to **-e** and **-x** respectively, but treat overlapping changes (that is, changes that would be flagged ==== in the normal listing) differently. The overlapping lines from both files are inserted by the edit script, bracketed by <<<<<< and >>>>>> lines. The **-E** option is used by Revision Control System (RCS) Merge to ensure that overlapping changes in the merged files are preserved and brought to someone's attention.

-e Creates an edit script for use with the **ed** command to incorporate into *File1* all changes between *File2* and *File3* (that is, the changes that normally would be flagged ==== and ====3).

-x Produces an edit script to incorporate only changes flagged ====.

Examples

To list the differences among three files:

```
diff3 fruit.a fruit.b fruit.c
```

If `fruit.a`, `fruit.b`, and `fruit.c` contain the following data:

fruit.a	**fruit.b**	**fruit.c**
banana	apple	grape
grape	banana	grapefruit
kiwi	grapefruit	kiwi
lemon	kiwi	lemon
mango	orange	mango
orange	peach	orange
peach	pear	peach
pare		

then the output from the **diff3** command shows the differences between these files as follows. (The comments on the right do not appear in the output.)

```
====            All three files are different.
1:1,2c          Lines 1 and 2 of the first file, fruit.a
  banana
  grape
2:1,3c          Lines 1 through 3 of fruit.b
  apple
  banana
  grapefruit
3:1,2c          Lines 1 and 2 of fruit.c
  grape
  grapefruit
====2           The second file, fruit.b, is different.
1:4,5c          Lines 4 and 5 the same in fruit.a and fruit.c.
2:4a            To make fruit.b look same, add after line 4.
3:4,5c
  lemon
  mango
====            The first file, fruit.a, is different.
1:8c
  pare
2:7c            fruit.b line 7 and fruit.c line 8 are the same
  pear
3:7a
```

Files

/usr/bin/diff3	Indicates the **diff3** command.
/usr/lbin/diff3prog	Called by the **diff3** shell script.

Related Information

The **diff** command, **ed** command.

Files in *Operating system and device management* introduces you to files and the way you can work with them.

Input and output redirection in *Operating system and device management* describes how the operating system processes input and output.

---

diffmk Command

Purpose

Marks differences between files.

Syntax

diffmk [{ **-ab***X* | **-ae***X*] [**-b**] [**-cb***X* | **-ce***X*] [**-db***X* | **-de***X*] *File1 File2* [*File3*]

Description

The **diffmk** command compares the English-language file specified by the *File1* parameter with the file by the *File2* parameter. It then creates a third file that includes **.mc** requests (for creating change marks) for the **nroff** and **troff** commands. The *File1* and *File2* parameters specify the old and new versions, respectively, of the files. The **diffmk** command writes the newly created file to the *File3* parameter, if specified, or else to standard output. The *File3* file contains the lines of the *File2* file plus inserted formatter **.mc** requests. When the *File3* file is formatted, the changed or inserted text is marked by a | (vertical bar) at the right margin of each line. An * (asterisk) in the margin indicates that a line was deleted.

If the **DIFFMARK** environment variable is defined, it names a command string that the **diffmk** command uses to compare the files. (Normally, the **diffmk** command uses the **diff** command.) For example, to handle extremely large files better, you can set the **DIFFMARK** variable to `diff -h`.

Parameters

File1 Specifies an English-language file that is compared to the file specified by the *File2* parameter. The results of the comparison comprise the file specified by the *File3* parameter. *File1* is considered the ″old″ file.

File2 Specifies an English-language file that is compared to the file specified by the *File1* parameter. The results of the comparison comprise the file specified by the *File3* parameter. *File2* is considered the ″new″ file.

File3 Specifies a file that contains lines of the *File2* file and includes inserted formatter **.mc** requests for the **nroff** and **troff** commands. The contents of this file are the results of a comparison between the files specified by the *File1* and *File2* parameters. When formatted, the changed text is marked by a (|) vertical bar at the right margin of each line. An * (asterisk) indicates the line was deleted. If *File3* is not specified, the results of the comparison are written to standard input.

Flags

-ab*X* Uses *X* to mark where added lines begin.

-ae*X* Uses *X* to mark where added lines end.

-b Ignores differences that are only changes in tabs or spaces on a line.

-cb*X* Uses *X* to mark where changed lines begin.

-ce*X* Uses *X* to mark where changed lines end.

-db*X* Uses *X* to mark where deleted lines begin.

-de*X* Uses *X* to mark where deleted lines end.

Examples

1. To mark the differences between two versions of a text file, enter:

   ```
   diffmk chap1.old chap1 chap1.nroff
   ```

   This produces a copy of `chap1` containing **nroff** and **troff** change mark requests to identify text that has been added to, changed in, or deleted from `chap1.old`. This copy is saved in the `chap1.nroff` file.
2. To mark differences with non-**nroff** and **troff** messages, enter:

   ```
   diffmk -ab'>>New:' -ae'<<End New' \
   chap1.old chap1 chap1.nroff
   ```

   This causes the **diffmk** command to write `>>New:` on the line before a section of newly added lines to `chap1`, and to write `<<End New` on the line following the added lines. Changes and deletions still generate **nroff** and **troff** commands to put a | (vertical bar) or * (asterisk) in the margin.
3. To use different **nroff** and **troff** command-marking requests and ignore changes in white space, enter:

   ```
   diffmk -b -cb'.mc %' chap1.old chap1 chap1.nroff
   ```

   This imbeds commands that mark changes with `%` (percent sign) additions with a | (vertical bar), and deletions with an * (asterisk). It does not mark changes that only involve a different number of spaces or tabs between words (`-b`).

Related Information

The **diff** command, **nroff** command, **troff** command.

---

dig Command

Purpose

DNS lookup utility.

Syntax

dig [*@server*] [**-b** *address*] [**-c** *class*] [**-f** *filename*] [**-k** *filename*] [**-n**][**-p** *port#*] [**-t** *type*] [**-x** *addr*] [**-y** *name:key*] [*name*] [*type*] [*class*] [*queryopt...*]

dig [**-h**]

dig [*global-queryopt...*] [*query...*]

Description

The **dig** (domain information groper) command is a flexible tool for interrogating DNS name servers. It performs DNS lookups and displays the answers that are returned from queried name server(s). Most DNS administrators use **dig** to troubleshoot DNS problems because of its flexibility, ease of use, and clarity of output. Although **dig** is normally used with command-line arguments, it also has a batch mode for reading lookup requests from a file. Unlike earlier versions, the BIND9 implementation of **dig** allows multiple lookups to be issued from the command line. Unless it is told to query a specific name server, **dig** will try each of the servers listed in **/etc/resolv.conf**. When no command line arguments or options are given, **dig** will perform an NS query for ″**.**″ (the root).

Flags

-b *address* Sets the source IP address of the query-to address. This must be a valid address on one of the host's network interfaces.

-c *class* The default query class (IN for internet) is overridden by the **-c** option. ***class*** is any valid class, such as HS for Hesiod records or CH for CHAOSNET records.

-f *filename* Makes **dig** operate in batch mode by reading a list of lookup requests to process from the file ***filename***. The file contains a number of queries; one per line. Each entry in the file should be organized in the same way they would be presented as queries to **dig** using the command-line interface.

-h A brief summary of its command-line arguments and options is printed when the **-h** option is given.

-k *filename* To sign the DNS queries sent by **dig** and their responses using transaction signatures (TSIG), specify a TSIG key file using the **-k** option.

-n By default, IPv6 addresses are looked up using the IP6.ARPA domain and binary labels as defined in RFC2874. To use the older RFC1886 method using the IP6.INT domain and **nibble** labels, specify the **-n** (nibble) option.

-p *port#* If a non-standard port number is to be queried, the **-p** option is used. port# is the port number that dig will send its queries instead of the standard DNS port number 53. This option would be used to test a name server that has been configured to listen for queries on a non-standard port number.

-t *type* Sets the query type to ***type***. It can be any valid query type which is supported in BIND9. The default query type is **A**, unless the **-x** option is supplied to indicate a reverse lookup. A zone transfer can be requested by specifying a ***type*** of AXFR. When an incremental zone transfer (IXFR) is required, ***type*** is set to **ixfr=N**. The incremental zone transfer will contain the changes made to the zone since the serial number in the zone's SOA record was **N**.

-x *addr* Reverse lookups (mapping addresses to names) are simplified by the **-x** option. ***addr*** is an IPv4 address in dotted-decimal notation, or a colon-delimited IPv6 address. When this option is used, there is no need to provide the *name*, *class* and *type* arguments. **dig** automatically performs a lookup for a name like `11.12.13.10.in-addr.arpa` and sets the query type and class to PTR and IN respectively.

-y *name:key* You can also specify the TSIG key itself on the command line using the **-y** option; ***name*** is the name of the TSIG key and ***key*** is the actual key. The key is a base-64 encoded string, typically generated by **dnssec-keygen**(8). Caution should be taken when using the **-y** option on multi-user systems as the key can be visible in the output from ps(1) or in the shell's history file. When using TSIG authentication with **dig**, the name server that is queried needs to know the key and algorithm that is being used. In BIND, this is done by providing appropriate key and server statements in **named.conf**.

Parameters

global-queryopt... Global query option (see Multiple Querys).

query Query option (see Query Options).

Query Options

dig provides a number of query options which affect the way in which lookups are made and the results displayed. Some of these set or reset flag bits in the query header, some determine which sections of the answer get printed, and others determine the timeout and retry strategies. Each query option is identified by a keyword preceded by a plus sign (+). Some keywords set or reset an option. These may be preceded by the string no to negate the meaning of that keyword. Other keywords assign values to options like the timeout interval. They have the form **+keyword=value**. The query options are:

+[no]tcp
Use [do not use] TCP when querying name servers. The default behavior is to use UDP unless an AXFR or IXFR query is requested, in which case a TCP connection is used.

+[no]vc
Use [do not use] TCP when querying name servers. This alternate syntax to **+[no]tcp** is provided for backwards compatibility. The **vc** stands for virtual circuit.

+[no]ignore
Ignore truncation in UDP responses instead of retrying with TCP. By default, TCP retries are performed.

+domain=somename
Set the search list to contain the single domain ***somename***, as if specified in a domain directive in **/etc/resolv.conf**, and enable search list processing as if the **+search** option were given.

+[no]search
Use [do not use] the search list defined by the search list or domain directive in **resolv.conf** (if any). The search list is not used by default.

+[no]defname
Deprecated, treated as a synonym for **+[no]search**

+[no]aaonly
This option does nothing. It is provided for compatibility with old versions of **dig** where it sets an unimplemented resolver flag.

+[no]adflag
Set [do not set] the AD (authentic data) bit in the query. The AD bit currently has a standard meaning only in responses, not in queries, but the ability to set the bit in the query is provided for completeness.

+[no]cdflag
Set [do not set] the CD (checking disabled) bit in the query. This requests the server to not perform DNSSEC validation of responses.

+[no]recursive
Toggle the setting of the RD (recursion desired) bit in the query. This bit is set by default, which means dig normally sends recursive queries. Recursion is automatically disabled when the **+nssearch** or **+trace** query options are used.

+[no]nssearch
When this option is set, **dig** attempts to find the authoritative name servers for the zone containing the name being looked up and display the SOA record that each name server has for the zone.

+[no]trace
Toggle tracing of the delegation path from the root name servers for the name being looked up. Tracing is disabled by default. When tracing is enabled, **dig** makes iterative queries to resolve the name being looked up. It will follow referrals from the root servers, showing the answer from each server that was used to resolve the lookup.

+[no]cmd
Toggles the printing of the initial comment in the output identifying the version of **dig** and the query options that have been applied. This comment is printed by default.

+[no]short
Provide a terse answer. The default is to print the answer in a verbose form.

+[no]identify
Show [or do not show] the IP address and port number that supplied the answer when the **+short** option is enabled. If short form answers are requested, the default is not to show the source address and port number of the server that provided the answer.

+[no]comments
Toggle the display of comment lines in the output. The default is to print comments.

+[no]stats
This query option toggles the printing of statistics: when the query was made, the size of the reply and so on. The default behavior is to print the query statistics.

+[no]qr
Print [do not print] the query as it is sent. By default, the query is not printed.

+[no]question
Print [do not print] the question section of a query when an answer is returned. The default is to print the question section as a comment.

+[no]answer
Display [do not display] the answer section of a reply. The default is to display it.

+[no]authority
Display [do not display] the authority section of a reply. The default is to display it.

+[no]additional
Display [do not display] the additional section of a reply. The default is to display it.

+[no]all
Set or clear all display flags.

+time=T
Sets the timeout for a query to ***T*** seconds. The default time out is 5 seconds. An attempt to set ***T*** to less than 1 will result in a query timeout of 1 second being applied.

+tries=A
Sets the number of times to retry UDP queries to server to ***A*** instead of the default, 3. If ***A*** is less than or equal to zero, the number of retries is silently rounded up to 1.

+ndots=D
Set the number of dots that have to appear in name to ***D*** for it to be considered absolute. The default value is that defined using the **ndots** statement in **/etc/resolv.conf**, or 1 if no **ndots** statement is present. Names with fewer dots are interpreted as relative names and will be searched for in the domains listed in the search or domain directive in **/etc/resolv.conf**.

+bufsize=B
Set the UDP message buffer size advertised using EDNS0 to ***B*** bytes. The maximum and minimum sizes of this buffer are 65535 and 0 respectively. Values outside this range are rounded up or down appropriately.

+[no]multiline
Print records like the SOA records in a verbose multi-line format with human-readable comments. The default is to print each record on a single line, to facilitate machine parsing of the **dig** output.

Multiple Queries

The BIND 9 implementation of **dig** supports specifying multiple queries on the command line (in addition to supporting the **-f** batch file option). Each of those queries can be supplied with its own set of flags, options and query options.

In this case, each query argument represent an individual query in the command-line syntax described above. Each consists of any of the standard options and flags, the name to be looked up, an optional query type and class and any query options that should be applied to that query.

A global set of query options, which should be applied to all queries, can also be supplied. These global query options must precede the first tuple of name, class, type, options, flags, and query options supplied on the command line. Any global query options (except the **+[no]cmd** option) can be overridden by a query-specific set of query options. For example:

```
dig +qr www.isc.org any -x 127.0.0.1 isc.org ns +noqr
```

shows how **dig** could be used from the command line to make three lookups: an ANY query for **www.isc.org**, a reverse lookup of 127.0.0.1 and a query for the NS records of **isc.org**. A global query option of **+qr** is applied, so that **dig** shows the initial query it made for each lookup. The final query has a local query option of **+noqr** which means that **dig** will not print the initial query when it looks up the NS records for **isc.org**.

Examples

A typical invocation of **dig** looks like:

```
dig @server name type
```

where:

server The name or IP address of the name server to query. This can be an IPv4 address in dotted-decimal notation or an IPv6 address in colon-delimited notation. When the supplied server argument is a hostname, **dig** resolves that name before querying that name server. If no server argument is provided, **dig** consults **/etc/resolv.conf** and queries the name servers listed there. The reply from the name server that responds is displayed.

name The name of the resource record that is to be looked up.

type Indicates what type of query is required — **ANY**, **A**, **MX**, **SIG**, etc. type can be any valid query type. If no type argument is supplied, **dig** will perform a lookup for an ***A*** record.

Files

/etc/resolv.conf

Related Information

The host and dnssec-keygen commands.

The named8 daemon.

RFC1035

---

digest Command

Purpose

Converts the ASCII form of the **/etc/qconfig** file into the **/etc/qconfig.bin** file, a binary version of the queue configuration used by the **qdaemon** command. This command should not be entered on the command line; it is called by the **qdaemon** command.

Syntax

/usr/lib/lpd/digest *ASCIIFile BinaryFile*

Description

The **digest** command accepts an input file of ASCII characters and converts it into a binary file. This command is only used by the **qdaemon** command to translate the **/etc/qconfig** file into the binary version of the file, the **/etc/qconfig.bin** file.

Files

/etc/qconfig	Contains the queue configuration file.
/usr/sbin/qdaemon	Contains the queuing daemon.
/etc/qconfig.bin	Contains the digested, binary version of the **/etc/qconfig** file.

Related Information

The **qdaemon** command.

dircmp Command

Purpose

Compares two directories and the contents of their common files.

Syntax

dircmp [**-d**] [**-s**] [**-w** *num*] *Directory1 Directory2*

Description

The **dircmp** command compares the two directories specified by the *Directory1* and *Directory2* parameters and writes information about their contents to standard output. First, the **dircmp** command compares the file names in each directory. If the same file name appears in both, the **dircmp** command compares the contents of both files.

In the output, the **dircmp** command lists the files unique to each directory. It then lists the files with identical names in both directories, but with different contents. If no flag is specified, it also lists files that have identical contents as well as identical names in both directories.

The **diff -r** command offers a function similar to the **dircmp** command.

Flags

-d Displays for each common file name both versions of the differing file contents. The display format is the same as that for the **diff** command.
-s Does not list the names of identical files.
-w Change the width of the output to *num* number of characters.
num

Exit Status

This command returns the following exit values:

0 Successful completion.
>0 An error occurred.

Note: Differences in directory contents are not considered errors.

Examples

1. To summarize the differences between the files in two directories, type the following:

   ```
   dircmp proj.ver1 proj.ver2
   ```

   This displays a summary of the differences between the directories `proj.ver1` and `proj.ver2`. The summary lists separately the files found only in one directory or the other, and those found in both. If a file is found in both directories, the **dircmp** command notes whether the two copies are identical.
2. To show the details of the differences between files, type the following:

   ```
   dircmp  -d  -s proj.ver1 proj.ver2
   ```

   The **-s** flag suppresses information about identical files. The **-d** flag displays a **diff** listing for each of the differing files found in both directories.
3. To show the details of the differences between files with the width of the output line set to 90 characters, type the following:

```
$dircmp -w 90 dir1 dir2
```

Files

/usr/bin/dircmp Contains the **dircmp** command.

Related Information

The **cmp** command, **diff** command.

Directories in *Operating system and device management* describes the structure and characteristics of directories in the file system.

Input and output redirection in *Operating system and device management* describes how the operating system processes input and output.

---

dirname Command

Purpose

Writes to standard output all but the last part of a specified path.

Syntax

dirname *Path*

Description

The **dirname** command reads the specified path name, deletes all but the last / (slash) and the characters following it, and writes the result to standard output. If no characters follow the last /, the **dirname** command uses the next to last / and ignores all characters following it. The **dirname** command applies the following rules in creating the path name:

1. If the *Path* parameter is a // (double slash), or if the *Path* parameter consists entirely of slash characters, change the string to a single / (slash). Skip steps 2 through 7.
2. Remove any trailing / characters from the specified path.
3. If there are no / characters remaining in the *Path* parameter, change the path to a single **.** (period). Skip steps 4 through 7.
4. Remove any trailing, non-slash characters from the path.
5. If the remaining path is // (double slash), go to step 6.
6. Remove any trailing slash characters from the path.
7. If the remaining path is empty, change the path to a single /.

For example, entering:

```
dirname //
```

results in a single / (slash). Entering:

```
dirname /a/b/
```

results in /a. Entering:

```
dirname a
```

results in a single **.** (period). Entering:

```
dirname a/b
```

results in the path name `a`.

The **dirname** and **basename** commands are generally used inside command substitutions within a shell procedure to specify an output file name that is some variation of a specified input file name.

Exit Status

This command returns the following exit values:

0 Successful completion
>0 An error occurred.

Examples

To construct the name of a file located in the same directory as another, enter:

```
AOUTFILE=`dirname $TEXTFILE`/a.out
```

This sets the shell variable `AOUTFILE` to the name of an **a.out** file that is in the same directory as `TEXTFILE`. If `TEXTFILE` is **/home/fran/prog.c**, the value of `dirname $TEXTFILE` is **/home/fran** and `AOUTFILE` becomes **/home/fran/a.out**.

Files

/usr/bin/dirname	Contains the **dirname** command.

Related Information

The **basename** command, **sh** command.

---

disable Command

The **disable** command includes information for the AIX Print Subsystem **disable** and the System V Print Subsystem **disable**.

Purpose

Disables printer queue devices.

Syntax

disable [**-c**] [**-r***Reason*] *PrinterName ...*

Description

The **disable** command disables or brings offline the printer queue devices specified by the *PrinterName* parameter.

Note: You must have root user authority or belong to the printq group to run this command.

Flags

-c	Cancels all job requests. Using this flag is the same as entering the **enq -K** command.
-r*Reason*	Specifies the reason for disabling the printer queue device with the *Reason* variable. This flag is a "no operation" flag, which means that the system ignores this flag.

Security

Attention RBAC users and Trusted AIX users: This command can perform privileged operations. Only privileged users can run privileged operations. For more information about authorizations and privileges, see Privileged Command Database in *Security*. For a list of privileges and the authorizations associated with this command, see the **/etc/security/privcmds** database file.

Examples

1. To bring printer queue `lp0` offline without waiting for the current print jobs to finish, type:

   ```
   disable  -c lp0
   ```

2. To bring printer queue `lp0` offline after all print jobs are finished, type:

   ```
   disable lp0
   ```

Files

/usr/sbin/qdaemon	Queuing daemon
/etc/qconfig	Queue configuration file
/etc/qconfig.bin	Digested, binary version of the **/etc/qconfig** file
/var/spool/lpd/qdir/*	Queue requests
/var/spool/lpd/stat/*	Information on the status of the devices
/var/spool/qdaemon/*	Temporary copies of enqueued files

Related Information

The **cancel** command, **enable** command, **enq** command, **lp** command, **lpstat** command.

Starting and Stopping a Print Queue in *Printers and printing*.

System V Print Subsystem disable Command

Purpose

Disable LP printers

Syntax

disable [*flags*] *printers*

Description

The **disable** command deactivates the named *printers*, disabling them from printing requests submitted by **lp**. By default, any requests that are currently printing on the designated printers will be reprinted in their entirety either on the same printer or on another member of the same class of printers. If the printer is remote, this command will only stop the transmission of jobs to the remote system. The **disable** command must be run on the remote system to disable the printer. (Run **lpstat -p** to get the status of printers.)

Printer names are *system-defined words* and as such should be restricted to uppercase and lowercase ASCII characters.

If you enter `disable -?`, the system displays the command usage message and returns 0.

Flags

-c Cancel any requests that are currently printing on any of the designated printers. This flag cannot be used with the **-W** flag. If the printer is remote, the **-c** flag is ignored.

-r *reason*

Assign a *reason* for the disabling of the printers. This *reason* applies to all *printers* specified. This *reason* is reported by **lpstat -p**. *reason* must be enclosed in quotes if it contains blanks. The default reason is *unknown reason* for existing printers, and *new printer* for printers just added to the system but not yet enabled.

-W Wait until the request currently being printed is finished before disabling the specified printer.

This flag cannot be used with the **-c** flag. If the printer is remote, the **-W** flag will be silently ignored.

Security

Attention RBAC users and Trusted AIX users: This command can perform privileged operations. Only privileged users can run privileged operations. For more information about authorizations and privileges, see Privileged Command Database in *Security*. For a list of privileges and the authorizations associated with this command, see the **/etc/security/privcmds** database file.

Files

/var/spool/lp/*

References

The **lp** command, **lpstat** command.

---

diskusg Command

Purpose

Generates disk accounting data by user ID.

Syntax

diskusg [**-X**] [**-U** *MaxUsers*] [**-i** *FileListName*] [**-p** *File*] [**-u** *File*] [**-v**] { **-s** [*File ...*] | *FileSystem ...* }

Description

The **diskusg** command generates intermediate disk-accounting information from data in the files specified with the *File* or *FileSystem* parameters or from standard input. The **diskusg** command writes one record per user to standard output. This command is called by the **dodisk** command, which can be run under the **cron** daemon. The output is in the following format:

UID	Contains the numerical user ID of the user.
Login	Contains the login name of the user.
Blocks	Contains the total number of 512-byte disk blocks allocated to the user.

The output of this command becomes the input of the **acctdisk** command, which converts the information to a total accounting record. The total accounting record is merged with other total accounting records to produce the daily report.

If you specify the *FileSystem* parameter, the **diskusg** command reads the i-nodes of the specified file systems to generate the usage data. The *FileSystem* parameters must be the special file names of the file system devices. For example, use the **/dev/hd4** device instead of / (root) directory to generate usage data for the root file system.

If you specify the *File* parameter, the input must be in a **diskusg** output format.

For more information on disk usage, see the **acctdusg** command**.**

Note: This command is for local devices only.

Flags

-i *FileListName*	Ignores the data in the *FileListName* file system. The *FileListName* variable specifies a list of file system names separated by commas or enclosed within quotation marks.
-p *File*	Uses the password file specified by the *File* variable to generate login names. The default is the **/etc/passwd** file.
-s [*File*]	Combines all records from the input file(s) or from standard input into a single record. The input data is already in a **diskusg** output format.
-U *MaxUsers*	Sets the maximum number of users that can be processed by the **diskusg** command. You need to use this flag only if the number of users is greater than the default of 5000.
-u *File*	Writes a record to the specified *File* variable for each file that is charged to a user ID of no one. Each record consists of the special file name, the i-node number, and the user ID.
-v	Writes a list of all files that are charged to no one to the standard error output.
-X	Prints and processes all available characters for each user name instead of truncating to the first 8 characters.

Security

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

Attention RBAC users and Trusted AIX users: This command can perform privileged operations. Only privileged users can run privileged operations. For more information about authorizations and privileges, see Privileged Command Database in *Security*. For a list of privileges and the authorizations associated with this command, see the **/etc/security/privcmds** database file.

Examples

To generate daily disk-accounting information, add a line similar to the following to the **/var/spool/cron/crontab/root** file:

```
0 2 * * 4 /usr/sbin/acct/dodisk
```

This command tells the **cron** daemon to run the **dodisk** command at 2 a.m. (`02`) each Thursday (`4`). The **dodisk** command calls both the **diskusg** and **acctdisk** commands.

Note: To perform this example, you must have root authority.

Files

/usr/sbin/acct/diskusg	Contains the **diskusg** command.
/etc/passwd	Contains the basic attributes of users.

Related Information

The **acctdisk** command, **acctmerg** command, **dodisk** command, **runacct** command.

The **acct** subroutine.

The **acct** file format and **utmp** file format.

Accounting commands, System accounting, Setting up an accounting system in *Operating system and device management*.

dispgid Command

Purpose

Displays a list of all valid group names.

Syntax

dispgid

Description

The **dispgid** command can be used to display a list of all group names on the system (one name per line). There are no parameters for this command. The following files are accessed in read-only mode to retrieve the data:

- **/etc/passwd**
- **/etc/group**
- **/etc/security/user**
- **/etc/security/limits**
- **/etc/security/group**
- **/etc/security/environ**

Exit Status

0 The command completed successfully.

>0 An error occurred.

Examples

1. To list all the valid groups in the machine enter the **dispgid** command as follows:

   ```
   dispgid
   ```

   The output looks similar to the following:

   ```
   system
   staff
   bin
   sys
   adm
   uucp
   mail
   security
   cron
   printq
   audit
   ecs
   nobody
   usr
   perf
   ```

Files

/usr/sbin/dispgid	Contains the **dispgid** command
/etc/group	Contains group information

Related Information

The **dispuid** command, **lsgroup** command

---

dispuid Command

Purpose

Displays a list of all valid user names.

Syntax

dispuid

Description

This command can be used to display a list of all user names on the system (one line per name). There are no parameters for this command. The following files are accessed in read-only mode to retrieve the user data:

- **/etc/passwd**
- **/etc/security/user**
- **/etc/security/user.roles**
- **/etc/security/limits**
- **/etc/security/environ**
- **/etc/group**
- **/etc/group**

Exit Status

0 The command completed successfully.

>0 An error occurred.

Examples

1. To list all the valid users in your machine enter the dispuid command as follows:

   ```
   dispuid
   ```

   The output looks similar to the following:

   ```
   root
   daemon
   bin
   sys
   adm
   uucp
   guest
   nobody
   lpd
   invscout
   imnadm
   user1
   ```

Files

/usr/sbin/dispuid	Contains the dispuid command.
/etc/passwd	Contains password information.

Related Information

the **dispgid** command, **lsuser** command.

---

dist Command

Purpose

Redistributes a message to additional addresses.

Syntax

dist [**+** Folder] [**-nodraftfolder** | **-draftfolder +***Folder*] [*Message* | **-draftmessage** *Message*] [**-annotate** [**-inplace** | **-noinplace**] | **-noannotate**] [**-form** *FormFile*] [**-editor** *Editor* | **-noedit**] [**-nowhatnowproc** | **-whatnowproc** *Program*]

Description

The **dist** command provides an interface for redistributing existing messages to a new list of addresses. By default, the **dist** command copies the current message in the current folder to the *UserMHDirectory***/draft** file and starts an editor. To specify a message in the current folder other than the default, use the *Message* parameter.

Once started, the editor prompts you to enter values for each header field. The **dist** command uses the header format defined in the *UserMHDirectory***/distcomps** file. (If this file does not exist, the system uses the **/etc/mh/distcomps** file.) Since the body of the message is the message you are redistributing, do not fill in the body. To define a format file other than *UserMHDirectory***/distcomps** file, use the **-form** flag.

To change the default editor, use the **-editor** flag or define the `Editor:` entry in your **$HOME/.mh_profile** file.

Press the Ctrl-D key sequence to exit the editor. Upon exiting the editor, the **dist** command starts the Message Handler (MH) `What Now?` prompt. Press the Enter key to see a list of the available **whatnow** subcommands. These subcommands enable you to continue editing the message header, list the message header, direct the disposition of the message, or end the processing of the **dist** command.

> **Note:** A line of dashes or a blank line must be left between the header and the body of the message for the message to be identified when it is sent.

Redistributed messages consist of the original header and body appended to a new header. The **draft** file you edit using the **dist** command consists of header fields only. A copy of the original message with the new draft message is not automatically stored.

To annotate the original message with redistribution information, use the **-annotate** flag. This flag appends the original message with the `Resent:` field, and the current date and time.

Flags

-annotate Annotates the message being redistributed with the lines:

```
Resent: date
Resent: address
```

Since the **-annotate** flag is not preserved over multiple executions of the command, annotation is completed only if the message is sent directly from the **dist** command. The **-inplace** flag forces annotation to be done in place in order to preserve links to the annotated message.

-draftfolder *+Folder*
: Places the draft message in the specified folder. If **-draftfolder** *+Folder* flag is followed by a *Message* variable, it is the same as using the **-draftmessage** flag. If *+Folder* is not specified, the draft message is placed in *Current-Folder*.

-draftmessage *Message*
: Specifies a draft message. By default, the system creates a new draft message in the current folder. The draft message becomes the current message.

-editor *Editor*
: Specifies the initial editor for preparing the message for distribution.

+*Folder*
: Identifies the folder that contains the message to redistribute. If a folder is not specified, then *Current-Folder* is assumed.

-form *FormFile*
: Determines the message form. The **dist** command treats each line in the specified form file.

-help
: Lists the command syntax, available switches (toggles), and version information.

  Note: For MH, the name of this flag must be fully spelled out.

-inplace
: Forces annotation to be done in place in order to preserve links to the annotated message.

Message
: Identifies the message to redistribute. Use the following references to specify messages:

  Number
  : Number of the message.

  cur or . (period)
  : Current message. This is the default.

  first
  : First message in a folder.

  last
  : Last message in a folder.

  next
  : Message following the current message.

  prev
  : Message preceding the current message.

-noannotate
: Suppresses annotation. This flag is the default.

-nodraftfolder
: Places the draft in the *UserMHDirectory***/draft** file.

-noedit
: Suppresses the initial edit.

-noinplace
: Prevents annotation in place. This flag is the default.

-nowhatnowproc
: Suppresses interactive processing of the **dist** command. The **-nowhatnowproc** flag prevents any edit from occurring.

-whatnowproc *Program*
: Starts the specified program to guide you through the distribution tasks. If you specify the **whatnow** command as the *Program* variable, the **dist** command starts an internal **whatnow** procedure instead of a program with the file name **whatnow**.

Profile Entries

The following entries are entered in the *UserMHDirectory***/.mh_profile** file:

`Current-Folder:`
: Sets the default current folder.

`Draft-Folder:`
: Sets the default folder for drafts.

`Editor:`
: Sets the default editor.

`fileproc:`
: Specifies the program used to refile messages.

`Path:`
: Specifies the user's MH directory.

`whatnowproc:`
: Specifies the program used to prompt `What now?` questions.

Security

Attention RBAC users and Trusted AIX users: This command can perform privileged operations. Only privileged users can run privileged operations. For more information about authorizations and privileges,

see Privileged Command Database in *Security*. For a list of privileges and the authorizations associated with this command, see the **/etc/security/privcmds** database file.

Examples

1. To redistribute the current message from the current folder, enter:

   ```
   dist
   ```

   The system prompts you for the header field values. After entering a value, press the Enter key. To skip an entry, press the Enter key without entering a value. You must fill in the `Resent-to:` field. After completing the headers, do not modify the body of the text. Press the Ctrl-D key sequence to exit the editor. The system prompts you with:

   ```
   What now?
   ```

   Press the Enter key to see a list of available options. If you want to redistribute this message, enter `send.` Your message is redistributed to the new list of addresses.
2. To redistribute a message to a new list of addresses when a message draft exists, enter:

   ```
   dist
   ```

   The system responds with a message similar to the following:

   ```
   Draft "$HOME/Mail/draft" exists (43 bytes).
   Disposition? _
   ```

   To redistribute this draft, enter:

   ```
   replace
   ```

   The system prompts you for the header field values. After entering a value, press the Enter key. To skip an entry, press the Enter key without entering a value. You must fill in the `Resent-to:` field. After completing the headers, do not modify the body of the text. Press the Ctrl-D key sequence to exit the editor. The system prompts you with:

   ```
   What now?
   ```

   Press the Enter key to see a list of available options. If you want to redistribute the draft, enter `send.` Your message is redistributed to the new list of addresses.
3. To redistribute message `15` from the `schedules` folder, enter:

   ```
   dist  +schedules 15
   ```

   The system prompts you for the header field values. After entering a value, press the Enter key. To skip an entry, press the Enter key without entering a value. You must fill in the `Resent-to:` field. After completing the headers, do not modify the body of the text. Press the Ctrl-D key sequence to exit the editor. The system prompts you with:

   ```
   What now?
   ```

   Press the Enter key to see a list of available options. To redistribute the message, type `send` and press the Enter key.

Files

/etc/mh/distcomps	Contains the system default message format.
UserMHDirectory/**distcomps**	Contains the user's default message format.
UserMHDirectory/**draft**	Contains the current draft file.
/usr/bin/dist	Contains the executable form of the **dist** command.

Related Information

The **ali** command, **anno** command, **comp** command, **forw** command, **prompter** command, **refile** command, **repl** command, **send** command, **whatnow** command.

The **mh_alias** file, **mh_profile** file.

Mail applications in *Networks and communication management*.

---

dmadm Command

Purpose

Operates Network Data Administration Facility (NDAF) on the admin server.

Syntax

dmadm [**param=***val*]

Description

With corresponding parameters, the **dmadm** command sets default directories, timeout values, level of logging, security method used, Kerberos keytab path, Kerberos principal, and communication ports on the admin server within an NDAF domain.

Parameters

The **dmadm** command takes one or more of the following optional parameter values:

[**-rpc_timeout=***val*] Sets the timeout for an RPC connect or call. Default is 300 seconds.

[**-log_level=***val*] Sets the level of logging for the log files. The default is **notice**. Possible values include the following values:

- **critical**
- **error**
- **warning**
- **notice**
- **information**

[**-security=***val*] Sets the type of security method used. The default is `krb5`. Values include:

auth_sys
: For **uid/gid** authentication

krb5
: For Kerberos authentication

krb5i
: For Kerberos integrity authentication

krb5p
: For Kerberos privacy authentication

[**-krb5_principal=***val*] Sets the Kerberos principal used for the **kinit**.

[**-admin_port=***val*] Sets the **dmadm** port waiting for RPC of the **dmf** client. Default value is 28000.

[**-serv_port=***val*] Sets the **dms** port waiting for the **dmadm** RPC. Default value is 28001.

[**-ndaf_dir=***val*] Sets the base directory for NDAF. It contains default databases, logs, and directories for cells, dsets, and replicas. The default for the base directory is **/var/dmf**. Other defaults include the following directories:

- **${ndaf_dir}/log** for logs
- **${ndaf_dir}/admin** for admin databases

[**-ndaf_log_dir=***val*]	Sets the directory for log files. It is set to **${ndaf_dir}/log** by default.
[**-krb5_keytab=***val*]	Indicates the Kerberos keytab path. If you do not specify the parameter and the system resource controller (SRC) is not in use, the keytab is defined either by the KRB5_KTNAME environment variable, or by the default as specified in the **/etc/krb5/krb5.conf** file (when the KRB5_KTNAME variable is not set). If you do not specify the parameter but the SRC is in use, the keytab is always the default as specified in the **/etc/krb5/krb5.conf** file.
[**-admin_cb_port=***val*]	Sets the **dmadm** port waiting for the **dms** RPC callbacks. The default is `28002`.

Exit Status

0	The command completed successfully.
>0	An error occurred.

Examples

1. To start **dmadm** using SRC on the admin server, enter:
   ```
   startsrc -s dmadm
   ```
2. To start **dmadm** using SRC and specifying **auth_sys** security, enter:
   ```
   startsrc -a "-security=auth_sys" -s dmadm
   ```

Location

/usr/sbin/dmadm

Related Information

The “dmf Command,” “dms Command” on page 168, “dms_enable_fs Command” on page 171.

The **chndaf** command in *AIX Version 6.1 Commands Reference, Volume 1*.

The **mkndaf** command, **lsndaf** command in *AIX Version 6.1 Commands Reference, Volume 3*.

The **rmndaf** command in *AIX Version 6.1 Commands Reference, Volume 4*.

AIX Network Data Administration Facility

---

dmf Command

Purpose

Implements the Network Data Administration Facility (NDAF) Admin Client executable.

Syntax

dmf *verb object parameter flag*

Description

The **dmf** command implements the NDAF CLI, which is the program name for the NDAF (Network Data Administration Facility) Admin Client executable. NDAF is an AIX solution for centralized creation, placement, replication, ongoing management, and namespace federation of file system data across a network of machines.

The **dmf** command is the prefix for all CLI commands in NDAF. These commands follow a consistent structure: a common prefix, the actual name of the executable (**dmf**), a verb such as **create** or **delete**, the *object* to which the action is being applied, and any subsequent parameters (such as *names*). These parameters are position-dependent.

Objects

The following objects are used in conjunction with the **dmf** command.

admin	Represents the **dmadm** (data management administrator) daemon and is used to configure the admin server. One single object of this class can be created on a server running the **dmadm** daemon. A machine running the **dmadm** daemon must also be running the **dms** (data management server) daemon. When an **admin** object is created, a **server** object with the same name is also created.
server	Represents a **dms** (data management server) daemon in the system. Also sets default attributes for data hosted on this server and for general server configuration.
cell	Represents a cell. A cell is a unit of management and namespace, hosted by an admin server, but independent of all other cells hosted by that admin server. A cell contains its own namespace, consisting of dsets, and contains its own NDAF role-based security objects. A cell can be placed (made visible through an NFS export) on any server defined for the admin server that the cell is hosted on.
dset	Represents read or write data sets that consist of NFS exported data paths, including those hosted on local and clustered file systems. This object class creates dsets and manages their attributes and their visibility in the cell (**dmf mount** operations).
replica	Represents read-only copies of a dset, which can be distributed across multiple servers. This object class creates replicas and manages their attributes, placements, and their visibility in the cell (**dmf mount** operations).
role	Represents a set of privileges assigned to a set of NDAF principals to delegate object management rights.
status	Represents the status of the request issued by the admin server.

Verbs

The following verbs are used in conjunction with the **dmf** command.

	Objects creation and deletion
create	Creates a logical object.
destroy	Destroys an object and all its content.
	Existing objects inspection
enumerate	Lists logical objects that were created.
show	Shows the attributes of an object. If used with the **status** object, the **dmf** command with the **show** verb lists the state of previous **dmf** commands requests.
clear	Clears the requests history and can only be used with the **status** object.
	Objects attributes update
set	Sets the value of a non-list attribute for an object.
add_to	Adds a key/value item to a list-based attribute for an object.
remove_from	Removes a key/value item from a list-based attribute for an object.
	Data update and access
update	Can only be used for **replica** objects. Causes a replica and its clone locations to be refreshed with the content of the original source dset.
source	Changes the source dset of a replica.

master	Elects another replica location to become the master location. It is originally the first place were the replica was created. The master location cannot be moved.
place	Can only be used for **cell** objects, **replica** objects and **dset** objects. For **cell** objects, this action makes the cell available for NFS mount on the related server. For **replica** objects, this action creates an additional copy location on the related server. For **dset** objects, this action is used with the **-m** flag to handle data referrals to data residing outside the NDAF framework.
unplace	Can only be used for **cell** objects, **replica** objects and **dset** objects. For **cell** objects, this action makes the cell unavailable for NFS mount on the related server. For **replica** objects, this action removes a copy location on the related server. For **dset** objects, with the **–m** flag, this action removes the data referrals to data residing outside of the NDAF framework.
mount	Can only be used with **dset** objects and **replica** objects. Let this object appear in the cell federation namespace.
unmount	Can only be used with **dset** objects and **replica** objects. Hide this object in the cell federation namespace.
resolve	Finds the object, **dset** objects or **replica** objects, that corresponds to a path exported within the cell.
	Problem solving
repair	Unexports and re-exports the dsets of the selected server and fixes internal data representation.
validate	Checks the consistency of an object in the different internal objects databases on the admin and on the object's server.
check_adm	Detects and reports inconsistencies in the NDAF admin database.
check_serv	Detects and reports inconsistencies in the data server database.
check_adm_serv	Checks admin-data server database consistency.

Flags

-a	Identifies the admin server that a command can be sent to with an attached string parameter. The communication port can be added using a colon separator. By default, port 28000 is used.
-c	Identifies the container that holds the object that this command is issued to with an attached string parameter. The container is generally a cell, but it can also be a server.
-d	Launches a request that runs asynchronously. The command returns as soon as the requests are launched.
-e	At server creation, specifies that the object being created refers to an external NDAF server and is not actually running the NDAF data server.
-f	When used with the **destroy** or **unplace** verb, forces the command without prompting confirmation.
-m	Specifies that the data of the created dset will be managed outside NDAF (typical use of this is for cluster machines).
-o	With an attached string parameter, specifies the name of the object that this command is addressed to (a **dset** object, a **replica** object, or a **role** object).
-r	Causes the CLI to print to the console the UUID (identifier) assigned to the requests generated by the admin server. This is useful for tracking request completion with the **dmf show status** command.
-w	Specifies how long the CLI must wait for the asynchronous portion of an operation to complete before timing out (the default is 120 seconds). This flag takes a numeric parameter. The units are in seconds.

Note: When used with Kerberos principals, the name of the user running the **dmf** command is used as the Kerberos principal by default. If the **kinit** command is issued with a different principal name, the DMF_PRINCIPAL environment variable must be set to this principal name before launching the **dmf** command.

Detailed verbs description

Objects creation and deletion:

create

This verb takes the following syntax:

dmf create *object* [*params*]

The **create** verb creates a logical object. The address parameters specified with it must point at the container for the object. A variable number of parameters are required, depending on the type of object being created.

Parameters:

object Specifies the type of object created. Values include the following (other parameters depend on the object):

admin Requires the name to be given to the admin server as a parameter. This object parameter takes the following syntax:

```
dmf create admin name [-r] [-a admin_server]
```

where:

-a *admin_server*
Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.

name Specifies the name for the admin server to be created.

-r Prints the **uuid** (identifier) assigned to the request.

Note: Entering `dmf create admin my_admin` also creates the **my_admin** server object.

server Requires the name of the server, its DNS name or IP address and port. This object parameter takes the following syntax:

```
dmf create server name dns_target [-e] [-r] [-a admin_server]
```

where:

-a *admin_server*
Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.

dns_target
Specifies the DNS name or IP address of the server. The port can be added using a colon separator.

-e Specifies that the object is external to NDAF.

name Specifies the name for the data server to be created.

-r Prints the **uuid** (identifier) assigned to the request.

cell Requires the name to be given to the cell. This object parameter takes the following syntax:

```
dmf create cell name  [-w timeout] [-r] [-a admin_server]
```

where:

-a *admin_server*
Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.

name Specifies the name for the cell to be created.

-r Prints the **uuid** (identifier) assigned to the request.

-w *timeout*
Specifies how long the command can wait before completing.

dset Requires the name of the **dset**, the hosting server, and (optionally) local path on the server. This object parameter takes the following syntax:

```
dmf create dset name server [path] [-r] [-a admin_server]
[-c container]
```

where:

-a *admin_server*
Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.

-c *container*
Specifies the container (the cell name).

name Specifies the name for the dset to be created.

path Specifies the local path on the server. If the *path* parameter is omitted, the server puts the **dset** in its default pool. The path of the created **dset** object can then be inspected using the **dmf show dset** command.

-r Prints the **uuid** (identifier) assigned to the request.

server Specifies the server name.

replica Requires the name of the replica, the hosting server, and (optionally) local path on the server. This object parameter takes the following syntax:

```
dmf create replica name server [path] [-d | -w timeout] [-r]
[-a admin_server] [-c container] [-o object]
```

where:

-a *admin_server*
Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.

-c *container*
Specifies the container (the cell name).

-d Specifies that the command must be run asynchronously.

name Specifies the name for the replica to be created.

-o *object*
Specifies the name of the object that this command is addressed to.

path Specifies the local path on the server. If the *path* parameter is omitted, the server puts the replica in its default pool for replicas. The path of the created **replica** object can then be inspected using the **dmf show replica** command.

-r Prints the **uuid** (identifier) assigned to the request.

server Specifies the server name.

-w *timeout*
Specifies how long the command can wait before completing.

role Requires the name of the role to be created. This object parameter takes the following syntax:

```
dmf create role name  [-r] [-a admin_server] [-c container]
```

where:

-a *admin_server*
Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.

-c *container*
Specifies the container (the cell name).

-r Prints the **uuid** (identifier) assigned to the request.

destroy

This verb takes the following syntax:

dmf destroy *object* [*params*]

The **destroy** verb destroys an object and all its content. The objects that depend on that object are also destroyed. For example, if a dset is destroyed, all its content is destroyed. If a cell is destroyed, all of its dsets and replicas are destroyed. The *address* parameters point to the object that is to be destroyed.

Parameters:

object Specifies the type of object destroyed. Values include the following (other parameters depend on the object):

admin This object parameter takes the following syntax:

```
dmf destroy admin  [-r] [-f] [-a admin_server]
```

where:

-a *admin_server*
Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.

-f Forces the action without confirmation.

-r Prints the **uuid** (identifier) assigned to the request.

server This object parameter takes the following syntax:

```
dmf destroy server [-r] [-f] [-a admin_server] [-c container]
```

where:

-a *admin_server*
Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.

-c *container*
Specifies the server name.

-f Forces the action without confirmation.

-r Prints the **uuid** (identifier) assigned to the request.

cell This object parameter takes the following syntax:

```
dmf destroy cell [-r] [-f] [-a admin_server] [-c container]
```

where:

-a *admin_server*
Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.

-c *container*
Specifies the cell name.

-f Forces the action without confirmation.

-r Prints the **uuid** (identifier) assigned to the request.

dset This object parameter takes the following syntax:

```
dmf destroy dset [-r] [-f] [-a admin_server] [-c container]
[-o object]
```

where:

-a *admin_server*
Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.

-c *container*
Specifies the cell name.

-f Forces the action without confirmation.

-o *object*
Specifies the name of the object this command is addressed to.

-r Prints the **uuid** (identifier) assigned to the request.

replica This object parameter takes the following syntax:

```
dmf destroy replica [-r] [-f] [-a admin_server] [-c container]
[-o object]
```

where:

-a *admin_server*
Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.

-c *container*
Specifies the cell name.

-f Forces the action without confirmation.

-o *object*
Specifies the name of the object this command is addressed to.

-r Prints the **uuid** (identifier) assigned to the request.

role This object parameter takes the following syntax:

```
dmf destroy role [-r] [-f] [-a admin_server] [-c container]
[-o object]
```

where:

-a *admin_server*
Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.

-c *container*
Specifies the cell name.

-f Forces the action without confirmation.

-o *object*
Specifies the name of the object this command is addressed to.

-r Prints the **uuid** (identifier) assigned to the request.

Existing objects inspection:

enumerate

This verb takes the following syntax:

dmf enumerate *object selector* [*params*]

The **enumerate** verb obtains lists of objects within containers such as dsets within a cell. It takes a two-part parameter.

The *params* parameter, which is optional and can be omitted, is a filter consisting in a text match pattern using ? to match a single character and * to match multiple characters. This parameter is used to restrict the list to the objects matching the filters.

Object	Selector	Parameters

admin

cell	A list of cells created on that admin.
server	A list of servers depending on that admin.

This object parameter takes the following syntax:

```
dmf enumerate admin type [pattern] [-r] [-a admin_server]
```

where:

-a *admin_server*
Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.

pattern Optional matching text pattern. Valid values are **?** and *****.

-r Prints the **uuid** (identifier) assigned to the request.

type Specifies the type of objects to return. Valid values are **server**, **cell**, and **admin**.

server

dset	A list of dsets created on that server.
replica	A list of replicas created on that server.

This object parameter takes the following syntax:

```
dmf enumerate server type [pattern] [-r] [-a admin_server] [-c container]
```

where:

-a *admin_server*
Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.

-c *container*
Specifies the server name.

pattern Optional matching text pattern. Valid values are **?** and *****.

-r Prints the **uuid** (identifier) assigned to the request.

type Specifies the type of objects to return. Valid values are **dset** and **replica**.

cell

dset	A list of dsets that are part of that cell.
replica	A list of replicas that are part of that cell.
role	A list of roles that have been defined for that cell.

This object parameter takes the following syntax:

```
dmf enumerate cell type [pattern] [-r] [-a admin_server] [-c container]
```

where:

-a *admin_server*
Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.

-c *container*
Specifies the cell name.

pattern Optional matching text pattern. Valid values are **?** and *****.

-r Prints the **uuid** (identifier) assigned to the request.

type Specifies the type of objects to return. Valid values are **dset**, **replica**, and **role**.

dset

server A list of servers on which that dset has been placed.

This object parameter takes the following syntax:

```
dmf enumerate dset server [pattern] [-r] [-a admin_server]
[-c container] [-o object]
```

where:

-a *admin_server*
Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.

-c *container*
Specifies the cell name.

-o *object*
Specifies the name of the object this command is addressed to.

pattern Optional matching text pattern. Valid values are **?** and *****.

-r Prints the **uuid** (identifier) assigned to the request.

replica

server A list of servers on which that replica has been placed.

This object parameter takes the following syntax:

```
dmf enumerate replica server [pattern] [-r] [-a admin_server]
[-c container] [-o object]
```

where:

-a *admin_server*
Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.

-c *container*
Specifies the cell name.

-o *object*
Specifies the name of the object this command is addressed to.

pattern Optional matching text pattern. Valid values are **?** and *****.

-r Prints the **uuid** (identifier) assigned to the request.

show

This verb takes the following syntax:

dmf show *object* [*params*]

The **show** verb shows the attributes of an object. When used with the **status** object, this action lists the history of requests on the admin server.

Parameters:

object Specifies the type of object destroyed. Values include the following (other parameters depend on the object):

admin This object parameter takes the following syntax:

```
dmf show admin  [-r] [-a admin_server]
```

where:

-a *admin_server*
Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.

-r Prints the **uuid** (identifier) assigned to the request.

server This object parameter takes the following syntax:

```
dmf show server [-r] [-a admin_server] [-c container]
```

where:

-a *admin_server*
Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.

-c *container*
Specifies the server name.

-r Prints the **uuid** (identifier) assigned to the request.

cell This object parameter takes the following syntax:

```
dmf show cell [-r] [-a admin_server] [-c container]
```

where:

-a *admin_server*
Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.

-c *container*
Specifies the cell name.

-r Prints the **uuid** (identifier) assigned to the request.

dset This object parameter takes the following syntax:

```
dmf show dset [-r] [-a admin_server] [-c container] [-o object]
```

where:

-a *admin_server*
Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.

-c *container*
Specifies the cell name.

-o *object*
Specifies the name of the object that this command is addressed to.

-r Prints the **uuid** (identifier) assigned to the request.

replica This object parameter takes the following syntax:

```
dmf show replica [-r] [-a admin_server] [-c container] [-o object]
```

where:

-a *admin_server*
Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.

-c *container*
Specifies the cell name.

-o *object*
Specifies the name of the object that this command is addressed to.

-r Prints the **uuid** (identifier) assigned to the request.

role This object parameter takes the following syntax:

```
dmf show role [-r] [-a admin_server] [-c container] [-o object]
```

where:

-a *admin_server*
: Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.

-c *container*
: Specifies the cell name.

-o *object*
: Specifies the name of the object that this command is addressed to.

-r Prints the **uuid** (identifier) assigned to the request.

status This object parameter takes the following syntax:

```
dmf show status depth [-r] [-a admin_server]
```

where:

-a *admin_server*
: Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.

depth Specifies how many records to return.

-r Prints the **uuid** (identifier) assigned to the request.

clear

This verb takes the following syntax:

dmf clear status [**-r**] [**-a** *admin_server*]

where:

-a *admin_server*	Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.
-r	Prints the **uuid** (identifier) assigned to the request.

The **clear** verb clears the admin requests history when used with the **status** object. All history activity is then lost.

Objects attributes update:

set

This verb takes the following syntax:

dmf set *object key=value* [*params*]

The **set** verb sets the value of a non-list attribute for an object. These are single attributes, as distinguished from the **add_to** verb. The parameters of the **set** verb are a key/value pair.

Parameters:

key Depending on the object type, the *key* parameter can be one of the following values:

DmMinRpcPort
(applies to server) Specifies the minimum RPC port number to be used for data transfers between servers.

DmMaxRpcPort
(applies to server) Specifies the maximum RPC port number to be used for data transfers between servers.

DmDefaultRepPath
(applies to server) Specifies the local path that incoming replicas can be placed in for a server if no path is specified.

DmDefaultDsetPath
(applies to server) Specifies the local path on a server that new dsets can be created in if no path is specified.

DmDTAPort
(applies to server) Specifies the port on a data server that can be used for initiating data transfers from another server.

DmLogLevel
(applies to server & cell) Sets the level of logging for the log files. The default is notice. Possible values include the following:

- critical
- error
- warning
- notice
- information

DmLocsMax
(applies to cell & replica & dset) Specifies the maximum number of location referrals that can be returned to an NFS client for an object. The selected locations to return are based on the network affinity of the locations to the client.

DmCreateDs
(applies to role) If this value is set to 0, this role forbids dset creation. The default value is 1.

DmDestroyDs
(applies to role) If this value is set to 0, this role forbids dset destruction. The default value is 1.

DmModifyDs
(applies to role) If this value is set to 0, this role forbids dset modification. The default value is 1.

DmDuplicateDs
(applies to role) If this value is set to 0, this role forbids dset replication. The default value is 1.

DmCreateRole
(applies to role) If this value is set to 0, this role forbids role creation. The default value is 1.

DmDestroyRole
(applies to role) If this value is set to 0, this role forbids role destruction. The default value is 1.

DmModifyRole
(applies to role) If this value is set to 0, this role forbids role modification. The default value is 1.

key (continued)

DmOwner
(applies to dset & replica) Specifies the owner that can be set to the dset root directory. The default value is root.

DmGroup
(applies to dset & replica) Specifies the group that can be set to the dset root directory. The default value is root.

DmMode
(applies to dset & replica) Specifies the mode that can be set to the dset root directory.

object
Specifies the type of object. Values include the following (other parameters depend on the object):

server
This object parameter takes the following syntax:

```
dmf set server key=value [-r] [-a admin_server] [-c container]
```

where:

-a *admin_server*
Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.

-c *container*
Specifies the server name.

key=*value*
Specifies an attribute and the value to assign to it. Valid keys are: **DmMinRpcPort**, **DmMaxRpcPort**, **DmDefaultRepPath**, **DmDefaultDsetPath**, **DmDTAPort**, **DmLogLevel**.

-r Prints the **uuid** (identifier) assigned to the request.

cell
This object parameter takes the following syntax:

```
dmf set cell key=value [-r] [-a admin_server] [-c container]
```

where:

-a *admin_server*
Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.

-c *container*
Specifies the cell name.

key=*value*
Specifies an attribute and the value to assign to it. Valid keys are: **DmLogLevel**, **DmLocsMax**.

-r Prints the **uuid** (identifier) assigned to the request.

dset This object parameter takes the following syntax:

```
dmf set dset key=value [-r] [-a admin_server]
[-c container] [-o object]
```

where:

-a *admin_server*
Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.

-c *container*
Specifies the cell name.

key=*value*
Specifies an attribute and the value to assign to it. Valid keys are: **DmOwner**, **DmGroup**, **DmMode**, **DmLocsMax**.

-o *object*
Specifies the name of the object this command is addressed to.

-r Prints the **uuid** (identifier) assigned to the request.

replica This object parameter takes the following syntax:

```
dmf set replica key=value [-r] [-a admin_server]
[-c container] [-o object]
```

where:

-a *admin_server*
Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.

-c *container*
Specifies the cell name.

key=*value*
Specifies an attribute and the value to assign to it. Valid keys are: **DmOwner**, **DmGroup**, **DmMode**, **DmLocsMax**.

-o *object*
Specifies the name of the object this command is addressed to.

-r Prints the **uuid** (identifier) assigned to the request.

role This object parameter takes the following syntax:

```
dmf set role key=value [-r] [-a admin_server]
[-c container] [-o object]
```

where:

-a *admin_server*
Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.

-c *container*
Specifies the cell name.

key=*value*
Specifies an attribute and the value to assign to it. Valid keys are: **DmCreateDs**, **DmDestroyDs**, **DmModifyDs**, **DmDuplicateDs**, **DmCreateRole**, **DmDestroyRole**, **DmModifyRole**.

-o *object*
Specifies the name of the object this command is addressed to.

-r Prints the **uuid** (identifier) assigned to the request.

add_to

This verb takes the following syntax:

dmf add_to *object key=value* [*params*]

The **add_to** verb adds a key/value item to a list-based attribute for an object. The additional parameters of the **add_to** verb specify which NDAF object is impacted.

Parameters:

key Depending on the object type, the *key* parameter can be one of the following values:

DmPrincipal
It is used to change the list of users that are the owners of this NDAF object. By default, the user that created the object is the only DmPrincipal.

DmClientDnsName
This is used for server objects. If a dset or a replica has a location on this server, this DnsName will appear in the locations list of this dset or this replica when requested through NFSv4.

DmOwningRole
It defines which roles control access rights delegation for this object.

DmTransferTable
It defines the list of protocols that can be used for data transfers between dsets and replica locations. By default, a standard copy protocol is used. The rsync protocol can also be added to the list. The protocol is negotiated during data transfers depending on the capabilities of the source and the target. The valid **DmTransferTable** list values are copy and rsync.

object Specifies the type of object. Values include the following (other parameters depend on the object):

admin This object parameter takes the following syntax:

```
dmf add_to admin key=value [-r] [-a admin_server]
```

where:

-a *admin_server*
Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.

key=value
Specifies an attribute and the value to assign to it. Valid key is **DmPrincipal**.

-r Prints the **uuid** (identifier) assigned to the request.

server This object parameter takes the following syntax:

```
dmf add_to server key=value [-r] [-a admin_server] [-c container]
```

where:

-a *admin_server*
Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.

-c *container*
Specifies the server name.

key=value
Specifies an attribute and the value to assign to it. Valid keys are **DmPrincipal**, **DmClientDnsName**, and **DmTransferTable**.

-r Prints the **uuid** (identifier) assigned to the request.

cell This object parameter takes the following syntax:

```
dmf add_to cell key=value [-r] [-a admin_server] [-c container] ]
```

where:

-a *admin_server*
Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.

-c *container*
Specifies the cell name.

key=value
Specifies an attribute and the value to assign to it. Valid key is **DmPrincipal**.

-r Prints the **uuid** (identifier) assigned to the request.

dset This object parameter takes the following syntax:

```
dmf add_to dset key=value [-r] [-a admin_server]
[-c container] [-o object]
```

where:

-a *admin_server*
Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.

-c *container*
Specifies the cell name.

key=value
Specifies an attribute and the value to assign to it. Valid keys are: **DmPrincipal**, **DmOwningRole**, **DmTransferTable**.

-o *object*
Specifies the name of the object this command is addressed to.

-r Prints the **uuid** (identifier) assigned to the request.

replica This object parameter takes the following syntax:

```
dmf add_to replica key=value [-r] [-a admin_server]
[-c container] [-o object]
```

where:

-a *admin_server*
Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.

-c *container*
Specifies the cell name.

key=value
Specifies an attribute and the value to assign to it. Valid keys are: **DmPrincipal**, **DmOwningRole**, **DmTransferTable**.

-o *object*
Specifies the name of the object this command is addressed to.

-r Prints the **uuid** (identifier) assigned to the request.

role This object parameter takes the following syntax:

```
dmf add_to role key=value [-r] [-a admin_server]
[-c container] [-o object]
```

where:

-a *admin_server*
Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.

-c *container*
Specifies the cell name.

key=*value*
Specifies an attribute and the value to assign to it. Valid keys are: **DmPrincipal**, **DmOwningRole**, **DmServer**, **DmMember**.

-o *object*
Specifies the name of the object this command is addressed to.

-r Prints the **uuid** (identifier) assigned to the request.

remove_from

This verb takes the following syntax:

dmf remove_from *object key=value* [*params*]

The **remove_from** verb removes a key/value item from a list-based attribute for an object. The additional parameters of the **remove_from** verb specify which NDAF object is impacted.

The **dmf show** command can be used to inspect the attributes lists.

Parameters:

key Depending on the object type, the key can be one of the following values:

DmPrincipal
It is used to change the list of users that are the owners of this NDAF object. By default, the user that created the object is the only DmPrincipal.

DmClientDnsName
This is used for server objects. If a dset or a replica has a location on this server, this DnsName will appear in the locations list of this dset or this replica when requested through NFSv4.

DmOwningRole
It defines which roles control access rights delegation for this object.

DmTransferTable
It defines the list of protocols that can be used for data transfers between dsets and replica locations. By default, a standard copy protocol is used. The rsync protocol can also be added to the list. The protocol is negotiated during data transfers depending on the capabilities of the source and the target.

object Specifies the type of object. Values include the following (other parameters depend on the object):

admin This object parameter takes the following syntax:

```
dmf remove_from admin key=value [-r] [-a admin_server]
```

where:

-a *admin_server*
Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.

key=*value*
Specifies an attribute and the value to assign to it. Valid key is **DmPrincipal**.

-r Prints the **uuid** (identifier) assigned to the request.

server This object parameter takes the following syntax:

```
dmf remove_from server key=value [-r] [-a admin_server]
[-c container]
```

where:

-a *admin_server*
Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.

-c *container*
Specifies the server name.

key=*value*
Specifies an attribute and the value to assign to it. Valid keys are **DmPrincipal**, **DmClientDnsName**, and **DmTransferTable**.

-r Prints the **uuid** (identifier) assigned to the request.

cell This object parameter takes the following syntax:

```
dmf remove_from cell key=value [-r] [-a admin_server]
[-c container]
```

where:

-a *admin_server*
Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.

-c *container*
Specifies the cell name.

key=*value*
Specifies an attribute and the value to assign to it. Valid key is **DmPrincipal**.

-r Prints the **uuid** (identifier) assigned to the request.

dset This object parameter takes the following syntax:

```
dmf remove_from dset key=value [-r] [-a admin_server]
[-c container] [-o object]
```

where:

-a *admin_server*
Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.

-c *container*
Specifies the cell name.

key=*value*
Specifies an attribute and the value to assign to it. Valid keys are: **DmPrincipal**, **DmOwningRole**, **DmTransferTable**.

-o *object*
Specifies the name of the object this command is addressed to.

-r Prints the **uuid** (identifier) assigned to the request.

replica This object parameter takes the following syntax:

```
dmf remove_from replica key=value [-r] [-a admin_server]
[-c container] [-o object]
```

where:

-a *admin_server*
Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.

-c *container*
Specifies the cell name.

key=*value*
Specifies an attribute and the value to assign to it. Valid keys are: **DmPrincipal**, **DmOwningRole**, **DmTransferTable**.

-o *object*
Specifies the name of the object this command is addressed to.

-r Prints the **uuid** (identifier) assigned to the request.

role This object parameter takes the following syntax:

```
dmf remove_from role key=value [-r] [-a admin_server]
[-c container] [-o object]
```

where:

-a *admin_server*
Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.

-c *container*
Specifies the cell name.

key=*value*
Specifies an attribute and the value to assign to it. Valid keys are: **DmPrincipal**, **DmOwningRole**, **DmServer**, **DmMember**.

-o *object*
Specifies the name of the object this command is addressed to.

-r Prints the **uuid** (identifier) assigned to the request.

Data update and access:

update

This verb takes the following syntax:

dmf update replica [**-d** | **-w** *timeout*] [**-r**] [**-a** *admin_server*] [**-c** *container*] [**-o** *object*]

where:

-a *admin_server*	Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.
-c *container*	Specifies the cell name.
-d	Specifies that the command must be run asynchronously.
-o *object*	Specifies the name of the object this command is addressed to.
-r	Prints the **uuid** (identifier) assigned to the request.
-w *timeout*	Specifies how long the command can wait before completing.

The **update** verb causes a replica and its clone locations to be refreshed with the content of the original source dset.

Note: The copy or the rsync protocol can be used for data transfers during the **update** operation. See the 153 field in the **add_to** section to specify which protocol must be used.

source

This verb takes the following syntax:

dmf source replica *source_dset* [**-r**] [**-a** *admin_server*] [**-c** *container*] [**-o** *object*]

where:

-a *admin_server*	Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.
-c *container*	Specifies the cell name.
-o *object*	Specifies the name of the object this command is addressed to.
source_dset	Specifies the dset that becomes the new source of the replica.
-r	Prints the **uuid** (identifier) assigned to the request.

The **source** verb changes the source dset of a replica.

master

This verb takes the following syntax:

dmf master replica *server* [*path*] [**-r**] [**-a** *admin_server*] [**-c** *container*] [**-o** *object*]

where:

-a *admin_server*	Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.
-c *container*	Specifies the cell name.
-o *object*	Specifies the name of the object this command is addressed to.
path	Specifies the path of the replica location.
-r	Prints the **uuid** (identifier) assigned to the request.
server	Specifies the server name.

The **master** verb elects another replica location to become the master location. Use this in case you want to change the master location of the replica. The master location is the first location updated on any **update** command. The other locations are updated afterwards asynchronously. The master location cannot be moved. Moreover, the master location is used to transmit data updates to the other locations. It must, therefore, be carefully selected to provide good data transfer performance.

place

This verb takes the following syntax:

The **place** verb places an object on a server. It can be applied to a cell, a replica, or a dset. Its parameters depend on the type of object. The action is different between a cell and a replica or dset.

For a cell, the **place** verb is used to make the cell visible through a server. The cell is exported under the server's **nfsroot** and contains referrals to mounted dsets and replicas. The parameter of the **dmf place cell** command is the name of the server. The root namespace of the cell is then placed on the server.

For a replica, the **place** verb creates a clone of the replica at the specified location on the server. If the replica is mounted in the cell, a referral to this clone location will be added to the referrals list returned to the NFS clients. The order of the referrals in this list depends on the network affinities. Every clone location of a replica is updated asynchronously upon update action requests. The **dmf place replica** command takes the server and optionally the local path on the server as parameters. For example:

```
dmf place replica my_server local_path -a my_admin -c my_cell -o my_replica
```

For a dset, the **place** verb is used in cluster file system environments, such as GPFS™, to provide the same coherent view of the dset through different servers in the cluster. The **-m** flag must be specified. No NDAF management or actions are carried out on the target dset. For example:

```
dmf place dset my_external_server external_server_path -m -a my_admin -c my_cell -o my_dset
```

The **place dset** action can be used only in cluster file system environments, such as GPFS, where the coherence view of the underlying data is assumed by the system and not by NDAF.

Parameters:

cell This object parameter takes the following syntax:

```
dmf place cell server  [-r] [-a admin_server] [-c container]
```

where:

-a *admin_server*
: Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.

-c *container*
: Specifies the cell name.

-r Prints the **uuid** (identifier) assigned to the request.

server Specifies the server name.

dset This object parameter takes the following syntax:

```
dmf place dset server  [path] [-d | -w timeout] [-r] [-m] [-a admin_server]
[-c container] [-o object]
```

where:

-a *admin_server*
Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.

-c *container*
Specifies the cell name.

-o *object*
Specifies the name of the object that this command is addressed to.

-d Specifies that the command must be run asynchronously.

-m Specifies that the data of the created dset will be managed outside NDAF (used for cluster machines).

path Specifies the path where the data resides.

-r Prints the **uuid** (identifier) assigned to the request.

server Specifies the server name.

-w *timeout*
Specifies how long the command can wait before completing.

replica This object parameter takes the following syntax:

```
dmf place replica server  [path] [-d | -w timeout] [-r] [-a admin_server]
[-c container] [-o object]
```

where:

-a *admin_server*
Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.

-c *container*
Specifies the cell name.

-o *object*
Specifies the name of the object that this command is addressed to.

-d Specifies that the command must be run asynchronously.

path Specifies the path where the data will reside.

-r Prints the **uuid** (identifier) assigned to the request.

server Specifies the server name.

-w *timeout*
Specifies how long the command can wait before completing.

unplace

This verb takes the following syntax:

Both cells and replicas can be unplaced from a server. The action differs depending on whether **dmf unplace** is used on a cell, dset or replica.

For **cell** objects, **unplace** actions prevent NFS users from mounting the cell on that server.

dmf unplace cell *server* [**-r**] [**-f**] [**-a** *admin_server*] [**-c** *container*]

where:

-a *admin_server*	Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.
-c *container*	Specifies the cell name.
-f	Forces the action without confirmation.
-r	Prints the **uuid** (identifier) assigned to the request.
server	Specifies the server name.

The **unplace cell** verb removes a cell from a server. It does not remove the data from the dsets or replicas and does not unexport the path to this data on the server named in the command line. Data will remain accessible to other clients on this server. The **unplace cell** verb will unexport the cell through NFS on that server. It remains visible on the other servers.

For **dset** objects, **unplace** actions will remove a referral to data that resides outside NDAF.

dmf unplace dset *server* [path] [**-r**] [**-f**] [**–m**] [**-a** *admin_server*] [**-c** *container*] [**–o** *object*]

where:

-a *admin_server*	Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.
-c *container*	Specifies the cell name.
-f	Forces the action without confirmation.
-m	Specifies that the data of the created dset will be managed outside NDAF (used for cluster machines).
-o *object*	Specifies the name of the object this command is addressed to.
path	Specifies the path where the location data resides and will be removed.
-r	Prints the **uuid** (identifier) assigned to the request.
server	Specifies the server name.

The **unplace dset** verb removes a referral to data that resides outside NDAF. The **unplace dset** verb does not remove the data from the dsets or replicas and does not unexport the path to this data on the server named in the command line. Data will remain accessible to other clients on this server.

For replicas, **unplace** actions will remove a location of the replica.

dmf unplace replica *server* [*path*] [**-r**] [**-f**] [**-a** *admin_server*] [**-c** *container*] [**–o** *object*]

where:

-a *admin_server*	Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.
-c *container*	Specifies the cell name.
-o *object*	Specifies the name of the object this command is addressed to.
-f	Forces the action without confirmation.
path	Specifies the path where the location data resides and will be removed.
-r	Prints the **uuid** (identifier) assigned to the request.
server	Specifies the server name.

For replicas, the **unplace** verb removes the clone location of the replica on that server. This location is unexported and its content is destroyed. The referral to this location is removed from the referrals list returned by NFS for this replica in the cell. The other locations of the replica remain the same. The first replica location (created with the **create** command) is called the *master location* of the replica. This location cannot be unplaced unless another location is elected as master location to replace the first one (see the master verb for more information).

If the replica on the server only has one location, specification of the path is not mandatory. If several paths are available, an error message will be returned.

Notes:

1. A cell cannot be unplaced from the admin.
2. The last location of a replica cannot be unplaced. Use the **destroy** verb instead.

mount

This verb takes the following syntax:

dmf mount *object* [*params*]

The **mount** verb mounts a dset or a replica in the federation namespace and makes it visible in the cell to NFS clients. In practice, an NFSv4 referral to the dset (exported in NFSv4 at creation time) or replica is added in the cell.

Parameters:

object Specifies the type of object created. Values include the following (other parameters depend on the object):

dset This object parameter takes the following syntax:

```
dmf mount dset mount_path [-r] [-a admin_server] [-c container]
 [-o object]
```

where:

-a *admin_server*
Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.

-c *container*
Specifies the cell name.

mount_path
Specifies the mount path in the namespace.

-o *object*
Specifies the name of the object this command is addressed to.

-r Prints the **uuid** (identifier) assigned to the request.

replica This object parameter takes the following syntax:

```
dmf mount replica mount_path [-r] [-a admin_server] [-c container]
[-o object]
```

where:

-a *admin_server*
Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.

-c *container*
Specifies the cell name.

mount_path
Specifies the mount path in the namespace.

-o *object*
Specifies the name of the object this command is addressed to.

-r Prints the **uuid** (identifier) assigned to the request.

unmount

This verb takes the following syntax:

dmf unmount *object* [*params*]

The unmount verb removes dset replica referrals from the cell federation namespace. They will consequently no longer appear in the cell for NFS clients. Its parameter is the existing mount point of the dset.

Note: If an NFS client has already resolved a referral, it will still be able to access this referral even after unmounting. If the referral was in a replicated dset, the replicas would have to be updated before the referral is removed as a result of the unmount.

Parameters:

object Specifies the type of object created. Values include the following (other parameters depend on the object):

dset This object parameter takes the following syntax:

```
dmf unmount dset mount_path [-r] [-a admin_server] [-c container]
[-o object]
```

where:

-a *admin_server*
Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.

-c *container*
Specifies the cell name.

mount_path
Specifies the mount path in the namespace.

-o *object*
Specifies the name of the object this command is addressed to.

-r Prints the **uuid** (identifier) assigned to the request.

replica This object parameter takes the following syntax:

```
dmf unmount replica mount_path [-r] [-a admin_server] [-c container]
[-o object]
```

where:

-a *admin_server*
Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.

-c *container*
Specifies the cell name.

mount_path
Specifies the mount path in the namespace.

-o *object*
Specifies the name of the object this command is addressed to.

-r Prints the **uuid** (identifier) assigned to the request.

resolve

This verb takes the following syntax:

dmf resolve cell *path* [**-r**] [**-a** *admin_server*] [**-c** *container*]

where:

-a *admin_server*	Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.
-c *container*	Specifies the cell name.
path	Specifies the path to look up.
-r	Prints the **uuid** (identifier) assigned to the request.

The **resolve** verb finds which dset or replica corresponds to a path within the cell. Its parameter is the path to look up (the given path is that of an NDAF mountpoint specified by the **dmf mount** command).

Problem solving:

repair

This verb takes the following syntax:

dmf repair server [**-r**] [**-V**] [**-U**] [**-R**] [**-a** <*admin_server*>] [**-c** <*container*>]

where:

-r	Prints the uuid (identifier) assigned to the request.
-V	Specifies that locations will be verified (default).
-U	Specifies that bad locations will be unexported.
-R	Specifies that bad locations will be repaired.
-a *value*	Specifies the DNS name or IP address of the admin server the port might be added using a colon separator.
-c *value*	Specifies the server name.

The **dmf repair** command provides dset and replica recovery on a data server. It is used to either restore a corrupted file system or migrate a data server to a different system if the NDAF dset data and state data is backed up.

The **-V** flag is used to check the dset and replica locations that are on the selected data server. This is the default action if no flag is set.

If the **-V** flag reports inconsistencies, the **-R** flag can be used to repair internal dset and replica descriptors. It will also reexport the dset and replica locations.

In case some locations are corrupted, you can request them to be unexported before doing the **repair (-R)** actions. To do so, run the **dmf repair** command with the **-U** (unmount) flag. Once the unexport action is done, you can destroy, recreate, and recover the file system from a backup. Finally, use the **dmf repair server** command with the **-R** option to recover the server and reexport the dsets within the file system.

validate

This verb takes the following syntax:

dmf validate *object* [*params*]

The **validate** verb checks the consistency of an object on the admin and on the object's server. A query is sent to the admin server daemon, which queries its database and the database of the object's server. Any consistent content found is returned.

Parameters:

object Specifies the type of object validated. Values include the following (other parameters depend on the object):

server This object parameter takes the following syntax:

```
dmf validate server [-r] [-a admin_server] [-c container]
```

where:

-a *admin_server*
Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.

-c *container*
Specifies the server name.

-r Prints the **uuid** (identifier) assigned to the request.

dset This object parameter takes the following syntax:

```
dmf validate dset [-r] [-a admin_server] [-c container]
[-o object]
```

where:

-a *admin_server*
Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.

-c *container*
Specifies the cell name.

-o *object*
Specifies the name of the object this command is addressed to.

-r Prints the **uuid** (identifier) assigned to the request.

replica This object parameter takes the following syntax:

```
dmf validate replica [-r] [-a admin_server] [-c container]
[-o object]
```

where:

-a *admin_server*
Specifies the DNS name or IP address of the admin server. The port can be added using a colon separator.

-c *container*
Specifies the cell name.

-o *object*
Specifies the name of the object this command is addressed to.

-r Prints the **uuid** (identifier) assigned to the request.

role This object parameter takes the following syntax:

```
dmf validate role [-r] [-f] [-a admin_server] [-c container]
[-o object]
```

where:

-a *admin_server*
Specifies the name for the data server to be created. The port can be added using a colon separator.

-c *container*
Specifies the cell name.

-f Forces the action without confirmation.

-o *object*
Specifies the name of the object this command is addressed to.

-r Prints the **uuid** (identifier) assigned to the request.

check_adm

This verb takes the following syntax:

dmf check_adm admin [**-a** *machine*]

The **check_adm** verb detects and reports inconsistencies in the NDAF admin database.

The tool compares each record and fills an error report each time a mismatch is encountered. As long as the client command has been executed correctly, the returned code will be 0. All other issues, such as communications problems between CLI and the admin server, will return a non-null error.

Note: The **check_adm** verb can not be used while other NDAF operations are running, because this can cause inaccurate report results.

check_serv

This verb takes the following syntax:

dmf check_serv server [**-a** *machine*] [**-c** *server*]

The **check_serv** verb detects and reports inconsistencies in the data server database.

The tool compares each record and fills an error report each time a mismatch is encountered. As long as the client command has been executed correctly, the returned code will be 0. All other issues, such as communications problems between CLI and the admin server, will return a non-null error.

Note: The **check_serv** verb can not be used while other NDAF operations are running, because this can cause inaccurate report results.

check_adm_serv

This verb takes the following syntax:

dmf check_adm_serv admin [**-a** *machine*] [**-c** *server*]

dmf check_adm_serv admin [**-a** *machine*]

Or

dmf check_adm_serv server [**-a** *machine*] [**-c** *server*]

The **check_adm_serv** verb checks admin-data server database consistency.

The tool compares each record and fills an error report each time a mismatch is encountered. As long as the client command has been executed correctly, the returned code will be 0. All other issues, such as communications problems between CLI and the admin server, will return a non-null error.

Note: The **check_adm_serv** verb can not be used while other NDAF operations are running, because this can cause inaccurate report results.

Exit Status

0	The command completed successfully.
>0	An error occurred.

Examples

1. To create an **admin** object and its linked data server on the host name where the **dms** and **dmadm** daemons run, enter:

   ```
   dmf create admin my_admin -a admin_host
   ```

2. To logically create a new server in the federation (to add a server to the federation) where the new server is called *server_name* and its DNS name is *server_dns_name*:

   ```
   dmf create server server_name server_dns_name -a admin_host
   ```

   The **dms** daemons must run on that machine.

3. To create a cell on the admin server, which will be the root of the namespace mounted by NFS clients, enter:

   ```
   dmf create cell my_cell -a admin_host
   ```

4. To create a dset within the cell where the dset is called *my_dset*, enter:

   ```
   dmf create dset my_dset server_name server_path -a admin_host -c my_cell
   ```

   The dset data will reside on *server_dns_name* in *server_path*.

5. To create a replica of the dset, enter:

   ```
   dmf create replica my_replica server_name replica_path -a admin_host -c my_cell -o my_dset
   ```

Location

/usr/bin/dmf

Related Information

The “dmadm Command” on page 137, “dms Command” on page 168, “dms_enable_fs Command” on page 171.

AIX Network Data Administration Facility

---

dmpuncompress Command

Purpose

Restores dump compressed files.

Syntax

/usr/bin/dmpuncompress [**-f**] [**-p**] [*File*]

Description

The **dmpuncompress** command restores original dump files that were compressed at dump time.

Each compressed file specified by the *File* parameter is removed and replaced by an expanded copy. The expanded file has the same name as the compressed version, but without the **.BZ** extension. If the user has root authority, the expanded file retains the same owner, group, modes, and modification time as the original file. If the user does not have root authority, the file retains the same modes and modification time, but acquires a new owner and group.

Flags

-f *File*	Forces expansion. Overwrites the file if it already exists. The system does not prompt the user that an existing file will be overwritten. File size might not actually shrink.
-p *File*	Preserves original **.BZ** file and uncompressed dump file. This overrides removal of the compressed file when there is a successful restoration of the original dump file. If restoration of the original dump file is incomplete because of an error, this option disables removal of the partial dump.

Exit Status

0	Successful completion.
>0	An error occurred.

Example

1. To uncompress the **dump.BZ** file, enter:

   ```
   /usr/bin/dmpuncompress dump.BZ
   ```

   The **dump.BZ** file is uncompressed and renamed **dump**.
2. To keep the **dump.BZ** file and the newly created dump file in the file system following completion, enter:

   ```
   /usr/bin/dmpuncompress -p dump.BZ
   ```

Location

/usr/bin/dmpuncompress

Related Information

The **savecore** command, the **snap** command, the **uncompress** command.

System Dump Facility in *AIX Version 6.1 Kernel Extensions and Device Support Programming Concepts*.

---

dms Command

Purpose

Operates AIX Network Data Administration Facility (NDAF) on a data server.

Syntax

dms [**param=***val*]

Description

With corresponding parameters, the **dms** command sets default directories, timeout values, level of logging, security method used, Kerberos keytab path, Kerberos principal, and communication ports on a data server within an NDAF domain.

Parameters

The **dms** command takes one or more of the following optional parameter values:

[**-rpc_timeout=***val*]
Sets the timeout for an RPC connect or call. Default is 300 seconds.

[**-log_level=***val*]
Sets the level of logging for the log files. The default is **notice**. Possible values include the following values:

- **critical**
- **error**
- **warning**
- **notice**
- **information**

[**-security=***val*]
Sets the type of security method used. The default is `krb5`. Values include:

auth_sys
For **uid/gid** authentication

krb5 For Kerberos authentication

krb5i For Kerberos integrity authentication

krb5p For Kerberos privacy authentication

[**-krb5_principal=***val*]
Sets the Kerberos principal used for the **kinit**.

[**-serv_port=***val*]
Sets the **dms** port waiting for the **dmadm** RPC. Default value is 28001.

[**-serv_serv_port=***val*]
Sets the **dms** port waiting for the other **dms** RPC. Default value is 28003.

[**-ndaf_dir=***val*]
Sets the base directory for NDAF. It contains default databases, logs, and directories for cells, dsets, and replicas. The default for the base directory is **/var/dmf**. Other defaults include the following directories:

- **${ndaf_dir}/log** for logs
- **${ndaf_dir}/server** for data server databases
- **${ndaf_dir}/dsets** for dsets, if the **-ndaf_dataset_default** parameter is not set
- **${ndaf_dir}/replicas** for replicas, if the **-ndaf_replica_default** parameter is not set

Note: At least either the **-ndaf_dataset_default** and **-ndaf_replica_default** parameters, or the **-ndaf_dir** parameter have to be specified. The creation of cells, datasets and replicas must have been enabled, using the **dms_enable_fs** command, on the file systems containing the specified directories to store the datasets and replicas.

[**-ndaf_dataset_default=***val*] Sets the default directory for dsets.
Note: At least either the **-ndaf_dataset_default** and **-ndaf_replica_default** parameters, or the **-ndaf_dir** parameter have to be specified. The creation of cells, datasets and replicas must have been enabled, using the **dms_enable_fs** command, on the file systems containing the specified directories to store the datasets and replicas.

[**-ndaf_replica_default=***val*] Sets the default directory for replicas.
Note: At least either the **-ndaf_dataset_default** and **-ndaf_replica_default** parameters, or the **-ndaf_dir** parameter have to be specified. The creation of cells, datasets and replicas must have been enabled, using the **dms_enable_fs** command, on the file systems containing the specified directories to store the datasets and replicas.

[**-ndaf_log_dir=***val*] Sets the directory for log files. It is set to **${ndaf_dir}/log** by default.

[**-krb5_keytab=***val*] Indicates the Kerberos keytab path. If you do not specify the parameter and the system resource controller (SRC) is not in use, the keytab is defined either by the KRB5_KTNAME environment variable, or by the default as specified in the **/etc/krb5/krb5.conf** file (when the KRB5_KTNAME variable is not set). If you do not specify the parameter but the SRC is in use, the keytab is always the default as specified in the **/etc/krb5/krb5.conf** file.

[**-admin_cb_port=***val*] Sets the **dmadm** port waiting for the **dms** RPC callbacks. The default is `28002`.

Exit Status

0 The command completed successfully.
>0 An error occurred.

Examples

1. To start the **dms** command using SRC and specifying the **/ndaf_fs** directory for NDAF files, enter:
   ```
   startsrc –a "-ndaf_dir=/ndaf_fs" –s dms
   ```
2. To start the **dms** command using SRC and specifying the **/ndaf_fs** directory for NDAF files and a security value of **auth_sys**, enter:
   ```
   startsrc -a "-ndaf_dir=/ndaf_fs \
    -security=auth_sys" -s dms
   ```

Location

/usr/sbin/dms

Related Information

The “dmadm Command” on page 137, “dmf Command” on page 138, “dms_enable_fs Command” on page 171.

The **chndaf** command.

The **mkndaf** command.

The **rmndaf** command.

The **lsndaf** command.

dms_enable_fs Command

Purpose

Enables, disables, or queries the capability to create Network Data Administration Facility (NDAF) cells, datasets, and replicas on a file system.

Syntax

dms_enable_fs [**-s** | **-u** | **-q**] *pathname* | **-h**

Description

The **dms_enable_fs** command enables, disables, or queries the capability to create cells, datasets, and replicas on a file system. It generates the **.DSETINFO** directory in the root of the file system. This directory must not be deleted.

Flags

-h	Displays usage of the **dms_enable_fs** command.
-q	Checks to see if VFS (path name within VFS) is enabled. If it is, 0 is returned. Otherwise, a nonzero value is returned.
-s	Enables a VFS (path name of VFS) for filesets.
-u	Disables a VFS (path name of VFS) for filesets.

Exit Status

0	The command completed successfully.
>0	An error occurred.

Examples

1. To enable the **/ndafexp** file system for datasets, enter:

```
dms_enable_fs -s /ndafexp
```

Location

/usr/sbin/dms_enable_fs

Related Information

The “dmadm Command” on page 137, “dmf Command” on page 138, “dms Command” on page 168.

AIX Network Data Administration Facility

dnssec-keygen Command

Purpose

DNSSEC key generation tool.

Syntax

dnssec-keygen [**-a** *algorithm*] [**-b** *keysize*] [**-n** nametype] [**-c** *class*] [**-e**] [**-g** *generator*] [**-h**] [**-p** *protocol*] [**-r** *randomdev*] [**-s** *strength*] [**-t** *type*] [**-v** *level*] [*name*]

Description

The **dnssec-keygen** command generates keys for DNSSEC (Secure DNS), as defined in RFC 2535. It can also generate keys for use with TSIG (Transaction Signatures), as defined in RFC 2845.

Flags

-a *algorithm*	Selects the cryptographic algorithm. The value of algorithm must be one of RSAMD5 or RSA, DSA, DH (Diffie Hellman), or HMAC-MD5. These values are case insensitive. Note that for DNSSEC, DSA is a mandatory to implement algorithm, and RSA is recommended. For TSIG, HMAC-MD5 is mandatory.
-b *keysize*	Specifies the number of bits in the key. The choice of key size depends on the algorithm used. RSA keys must be between 512 and 2048 bits. Diffie Hellman keys must be between 128 and 4096 bits. DSA keys must be between 512 and 1024 bits and an exact multiple of 64. HMAC-MD5 keys must be between 1 and 512 bits.
-n *nametype*	Specifies the owner type of the key. The value of nametype must either be ZONE (for a DNSSEC zone key), HOST or ENTITY (for a key associated with a host), or USER (for a key associated with a user). These values are case insensitive.
-c *class*	Indicates that the DNS record containing the key should have the specified class. If not specified, class IN is used.
-e	If generating an RSA key, use a large exponent.
-g *generator*	If generating a Diffie Hellman key, use this generator. Allowed values are 2 and 5. If no generator is specified, a known prime from RFC 2539 will be used if possible; otherwise the default is 2.
-h	Prints a short summary of the options and arguments to dnssec-keygen.
-p *protocol*	Sets the protocol value for the generated key. The protocol is a number between 0 and 255. The default is 2 (email) for keys of type USER and 3 (DNSSEC) for all other key types. Other possible values for this argument are listed in RFC 2535 and its successors.
-r *randomdev*	Specifies the source of randomness. If the operating system does not provide a **/dev/random** or equivalent device, the default source of randomness is keyboard input. randomdev specifies the name of a character device or file containing random data to be used instead of the default. The special value keyboard indicates that keyboard input should be used.
-s *strength*	Specifies the strength value of the key. The strength is a number between 0 and 15, and currently has no defined purpose in DNSSEC.
-t *type*	Indicates the use of the key. type must be one of AUTHCONF, NOAUTHCONF, NOAUTH, or NOCONF. The default is AUTHCONF. AUTH refers to the ability to authenticate data, and CONF the ability to encrypt data.
-v *level*	Sets the debugging level.

Generated Keys

When **dnssec-keygen** completes successfully, it prints a string of the form **Knnnn.+aaa+iiiii** to the standard output. This is an identification string for the key it has generated. These strings can be used as arguments to **dnssec-makekeyset**.

- **nnnn** is the key name.
- **aaa** is the numeric representation of the algorithm.
- **iiiii** is the key identifier (or footprint).

dnssec-keygen creates two files with names based on the printed string. **Knnnn.+aaa+iiiii.key** contains the public key, and **Knnnn.+aaa+iiiii.private** contains the private key. The **.key** file contains a DNS KEY record that can be inserted into a zone file (directly or with a $INCLUDE statement). The **.private** file contains algorithm specific fields. For obvious security reasons, this file does not have general read permission. Both **.key** and **.private** files are generated for symmetric encryption algorithm such as HMAC-MD5, even though the public and private key are equivalent.

Examples

To generate a 768-bit DSA key for the domain example.com, the following command would be issued:

```
dnssec-keygen -a DSA -b 768 -n ZONE example.com
```

The command would print a string of the form:

```
Kexample.com.+003+26160
```

In this example, **dnssec-keyge**n creates the files **Kexample.com.+003+26160.key** and **Kexample.com.+003+26160.private**.

Related Information

The **dnssec-makekeyset**, **dnssec-signkey**, and **dnssec-signzone**, commands

The BIND 9 Administrator Reference Manual.

RFC 2535, RFC 2845, and RFC 2539.

---

dnssec-makekeyset command

Purpose

DNSSEC zone signing tool.

Syntax

dnssec-makekeyset [**-a**] [**-s** *start-time*] [**-e** *end-time*] [**-h**] [**-p**] [**-r** *randomdev*] [**-t** *ttl*] [**-v** *level*] {*key...*}

Description

The **dnssec-makekeyset** command generates a key set from one or more keys created by **dnssec-keygen**. It creates a file containing a KEY record for each key, and self-signs the key set with each zone key. The output file is of the form **keyset-***nnnn***.**, where ***nnnn*** is the zone name.

Flags

-a	Verify all generated signatures.
-s *start-time*	Specify the date and time when the generated SIG records become valid. This can be either an absolute or relative time. An absolute start time is indicated by a number in YYYYMMDDHHMMSS notation; 20000530144500 denotes 14:45:00 UTC on May 30th, 2000. A relative start time is indicated by +N, which is N seconds from the current time. If no ***start-time*** is specified, the current time is used.
-e *end-time*	Specify the date and time when the generated SIG records expire. As with *start-time*, an absolute time is indicated in YYYYMMDDHHMMSS notation. A time relative to the start time is indicated with **+N**, which is N seconds from the start time. A time realtive to the current time is indicated with **now+N**. If no ***end-time*** is specified, 30 days from the start time is used as a default.
-h	Prints a short summary of the options and arguments to **dnssec-makekeyset**.
-p	Use pseudo-random data when signing the zone. This is faster, but less secure, than using real random data. This option may be useful when signing large zones or when the entropy source is limited.
-r *randomdev*	Specifies the source of randomness. If the operating system does not provide a **/dev/random** or equivalent device, the default source of randomness is keyboard input. randomdev specifies the name of a character device or file containing random data to be used instead of the default. The special value keyboard indicates that keyboard input should be used.
-t *ttl*	Specify the TTL (time to live) of the KEY and SIG records. The default is 3600 seconds.
-v *level Sets*	The debugging level.

Parameters

key The list of keys to be included in the keyset file. These keys are expressed in the form `Knnnn.+aaa+iiiii` as generated by **dnssec-keygen**.

Examples

The following command generates a keyset containing the DSA key for **example.com** generated in the **dnssec-keygen** man page.

```
dnssec-makekeyset -t 86400 -s 20000701120000 -e +2592000 Kexample.com.+003+26160
```

In this example, **dnssec-makekeyset** creates the file **keyset-example.com.**. This file contains the specified key and a self-generated signature. The DNS administrator for **example.com** could send **keyset-example.com.** to the DNS administrator for **.com** for signing, if the **.com** zone is DNSSEC-aware and the administrators of the two zones have some mechanism for authenticating each other and exchanging the keys and signatures securely.

Related Information

The **dnssec-keygen** and**dnssec-signkey** commands.

BIND 9 Administrator Reference Manual,

RFC 2535.

---

dnssec-signkey Command

Purpose

DNSSEC key set signing tool.

Syntax

dnssec-signkey [**-a**] [**-c** *class*] [**-s** *start-time*] [**-e** *end-time*] [**-h**] [**-p**] [**-r** *randomdev*] [**-v** *level*] *keyset key*

Description

The **dnssec-signkey** command signs a keyset. Typically the keyset will be for a child zone, and will have been generated by **dnssec-makekeyset**. The child zone's keyset is signed with the zone keys for its parent zone. The output file is of the form **signedkey-***nnnn*., where ***nnnn*** is the zone name.

Flags

-a Verify all generated signatures.

-c *class* Specifies the DNS class of the key sets.

-s *start-time* Specify the date and time when the generated SIG records become valid. This can be either an absolute or relative time. An absolute start time is indicated by a number in `YYYYMMDDHHMMSS` notation; 20000530144500 denotes 14:45:00 UTC on May 30th, 2000. A relative start time is indicated by **+N**, which is N seconds from the current time. If no *start-time* is specified, the current time is used.

-e *end-time* Specify the date and time when the generated SIG records expire. As with start-time, an absolute time is indicated in `YYYYMMDDHHMMSS` notation. A time relative to the start time is indicated with **+N**, which is N seconds from the start time. A time realtive to the current time is indicated with **now+N**. If no *end-time* is specified, 30 days from the start time is used as a default.

-h Prints a short summary of the options and arguments to **dnssec-signkey**.

-p Use pseudo-random data when signing the zone. This is faster, but less secure, than using real random data. This option may be useful when signing large zones or when the entropy source is limited.

-r *randomdev* Specifies the source of randomness. If the operating system does not provide a **/dev/random** or equivalent device, the default source of randomness is keyboard input. *randomdev* specifies the name of a character device or file containing random data to be used instead of the default. The special value keyboard indicates that keyboard input should be used.

-v *level* Sets the debugging level.

Parameters

keyset The file containing the child's keyset.

key The keys used to sign the child's keyset.

Examples

The DNS administrator for a DNSSEC-aware **.com** zone would use the following command to sign the keyset file for **example.com** created by **dnssec-makekeyset** with a key generated by **dnssec-keygen**:

```
dnssec-signkey keyset-example.com. Kcom.+003+51944
```

In this example, **dnssec-signkey** creates the file **signedkey-example.com.**, which contains the **example.com** keys and the signatures by the **.com** keys.

Related Information

The **dnssec-keygen**, **dnssec-makekeyset**, and **dnssec-signzone** commands.

---

dnssec-signzone Command

Purpose

DNSSEC zone signing tool.

Syntax

dnssec-signzone [**-a**] [**-c** *class*] [**-d** *directory*] [**-s** *start-time*] [**-e** *end-time*] [**-h**] [**-i** *interval*] [**-n** *nthreads*] [**-o** *origin*] [**-p**] [**-r** *randomdev*] [**-t**] [**-v** *level*] *zonefile key*...

...

Description

The **dnssec-signzone** command signs a zone. It generates NXT and SIG records and produces a signed version of the zone. If there is a **signedkey** file from the zone's parent, the parent's signatures will be incorporated into the generated signed zone file. The security status of delegations from the the signed zone (that is, whether the child zones are secure or not) is determined by the presence or absence of a **signedkey** file for each child zone.

Flags

-a Verify all generated signatures.

-c *class* Specifies the DNS class of the zone.

-d *directory* Look for **signedkey** files in directory as the directory.

-s *start-time* Specify the date and time when the generated SIG records become valid. This can be either an absolute or relative time. An absolute start time is indicated by a number in YYYYMMDDHHMMSS notation; 20000530144500 denotes 14:45:00 UTC on May 30th, 2000. A relative start time is indicated by **+N**, which is N seconds from the current time. If no *start-time* is specified, the current time is used.

-e *end-time* Specify the date and time when the generated SIG records expire. As with start-time, an absolute time is indicated in YYYYMMDDHHMMSS notation. A time relative to the start time is indicated with **+N**, which is N seconds from the start time. A time realtive to the current time is indicated with **now+N**. If no *end-time* is specified, 30 days from the start time is used as a default.

-f *output-file* The name of the output file containing the signed zone. The default is to **append .signed** to the input file.

-h Prints a short summary of the options and arguments to **dnssec-signzone**.

-i *interval* When a previously signed zone is passed as input, records may be resigned. The interval option specifies the cycle interval as an offset from the current time (in seconds). If a SIG record expires after the cycle interval, it is retained. Otherwise, it is considered to be expiring soon, and it will be replaced. The default cycle interval is one quarter of the difference between the signature end and start times. So if neither *end-time* or *start-time* are specified, **dnssec-signzone** generates signatures that are valid for 30 days, with a cycle interval of 7.5 days. Therefore, if any existing SIG records are due to expire in less than 7.5 days, they would be replaced.

-n *ncpus* Specifies the number of threads to use. By default, one thread is started for each detected CPU.

-o *origin* The zone origin. If not specified, the name of the zone file is assumed to be the origin.

-p Use pseudo-random data when signing the zone. This is faster, but less secure, than using real random data. This option may be useful when signing large zones or when the entropy source is limited.

-r *randomdev* Specifies the source of randomness. If the operating system does not provide a **/dev/random** or equivalent device, the default source of randomness is keyboard input. *randomdev* specifies the name of a character device or file containing random data to be used instead of the default. The special value keyboard indicates that keyboard input should be used.

-t Print statistics at completion.

-v *level* Sets the debugging level.

Parameters

zonefile The file containing the zone to be signed. Sets the debugging level.

key The keys used to sign the child's keyset.

Examples

The following command signs the **example.com** zone with the DSA key generated in the **dnssec-keygen** man page. The zone's keys must be in the zone. If there are **signedkey** files associated with this zone or any child zones, they must be in the current directory, **example.com**, the following command would be issued:

```
dnssec-signzone -o example.com db.example.com Kexample.com.+003+26160
```

In this example, **dnssec-signzone** creates the file **db.example.com.signed**. This file should be referenced in a zone statement in a **named.conf** file.

Related Information

The **dnssec-keygen**, **dnssec-makekeyset**, and **dnssec-signkey** commands.

BIND 9 Administrator Reference Manual

RFC 2535.

---

dodisk Command

Purpose

Initiates disk-usage accounting.

Syntax

/usr/sbin/acct/dodisk [**-X**] [**-o**] [*File ...*]

Description

The **dodisk** command initiates disk-usage accounting by calling the **diskusg** command and the **acctdisk** command. If you specify the **-o** flag with the **dodisk** command, a more thorough but slower version of disk accounting by login directory is initiated using the **acctdusg** command. Normally, the **cron** daemon runs the **dodisk** command.

By default, the **dodisk** command does disk accounting only on designated files with stanzas in the **/etc/filesystems** file and that contain the attribute **account=true**. If you specify file names with the *File* parameter, disk accounting is done on only those files.

If you do not specify the **-o** flag, the *File* parameter should contain the special file names of mountable file systems. If you specify both the **-o** flag and the *File* parameter, the files should be mount points of mounted file systems.

> **Note:** You should not share accounting files among nodes in a distributed environment. Each node should have its own copy of the various accounting files.

Flags

-o Calls the **acctdusg** command, instead of the **diskusg** command, to initiate disk accounting by login directory.
-X Process all available characters of each user name instead of truncating to the first 8 characters.

Security

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

Examples

1. To start automatic disk-usage accounting, add the following to the **/var/spool/cron/crontabs/root** file:

   ```
   0 2 * * 4 /usr/sbin/acct/dodisk
   ```

   This example shows the instructions that the **cron** daemon will read and act upon. The **dodisk** command will run at 2 a.m. (`0 2`) each Thursday (`4`). This command is only one of the accounting instructions normally given to the **cron** daemon. See ″Setting Up an Accounting System″ in *Operating system and device management* for more information on typical **cron** accounting entries.
2. To run disk-usage accounting on a system that contains user names greater than 8 character, add the following line to the **/var/spool/cron/crontabs/root** file:

   ```
   0 2 * * 4 /usr/sbin/acct/dodisk -X
   ```

Files

/usr/sbin/acct	The path to the accounting commands
/etc/filesystems	Contains information about file system.

Related Information

The **acctdisk** or **acctdusg** command, **diskusg** command.

The **cron** daemon.

For more information about the Accounting System, the preparation of daily and monthly reports, and the accounting files, see the System accounting in *Operating system and device management*.

Setting up an accounting system in *Operating system and device management* explains the steps you must take to establish an accounting system.

domainname Command

Purpose

Displays or sets the name of the current Network Information Service (NIS) domain.

Syntax

/usr/bin/domainname [*DomainName*]

Description

The **domainname** command displays or sets the name of the current NIS domain. If you do not specify a parameter, the **domainname** command displays the name of the current NIS domain. A domain typically encompasses a group of hosts under the same administration.

Only the root user can set the name of the domain by giving the **domainname** command an argument.

Security

Attention RBAC users and Trusted AIX users: This command can perform privileged operations. Only privileged users can run privileged operations. For more information about authorizations and privileges, see Privileged Command Database in *Security*. For a list of privileges and the authorizations associated with this command, see the **/etc/security/privcmds** database file.

Examples

1. To join a new domain, enter:

   ```
   domainname caesar
   ```

   In this example, the `domainname` command sets the NIS domain name to caesar.
2. To find out the name of the domain your machine belongs to, enter:

   ```
   domainname
   ```

Related Information

The **ypinit** command.

The **ypbind** daemon, **ypserv** daemon.

Network Information Service (NIS) Overview for System Management in *AIX Version 6.1 Network Information Services (NIS and NIS+) Guide*.

NIS Reference.

dosdel Command

Purpose

Deletes DOS files.

Syntax

dosdel [**-v**] [**-D** *Device*] *File ...*

Description

The **dosdel** command deletes the DOS file specified by the *File* parameter. Use the **-v** flag to obtain format information about the disk.

DOS file-naming conventions are used with one exception. Since the \ (backslash) character can have special meaning to the operating system, use a / (slash) character as the delimiter to specify subdirectory names in a DOS path name. The **dosdel** command converts lowercase characters in the file or directory name to uppercase before it checks the disk. Because all file names are assumed to be full (not relative) path names, you need not add the initial / (slash).

Flags

-D*Device*	Specifies the name of the DOS device as **/dev/fd0** or **/dev/fd1**. The default device is **/dev/fd0.**
-v	Writes information to standard output about the format of the disk. Use this flag to verify that a device is a DOS disk.

Examples

To delete a DOS file on the default device, enter:

```
dosdel file.ext
```

Files

/usr/bin/dosdel	Contains the **dosdel** command.

Related Information

The **dosdir** command, **dosformat** command, **dosread** command, **doswrite** command.

Files in *Operating system and device management* describes files, file types, and how to name files.

---

dosdir Command

Purpose

Lists the directory for DOS files.

Syntax

dosdir [**-l** [**-e**]] [**-a**] [**-d**] [**-t**] [**-v**] [**-D** *Device*] [*File ...* | *Directory ...*]

Description

The **dosdir** command displays information about the specified DOS files or directories. If you specify a directory without also specifying the **-d** flag, the **dosdir** command displays information about the files in that directory.

DOS file-naming conventions are used with one exception. Since the \ (backslash) character can have special meaning to the operating system, use a / (slash) character as the delimiter to specify subdirectory names in a DOS path name. The **dosdir** command converts lowercase characters in the file or directory name to uppercase before it checks the disk. Because all file names are assumed to be full (not relative) path names, you need not add the initial / (slash).

Flags

-a Writes information about all files. This includes hidden and system files as well as the . (dot) and .. (dot-dot) files.

-d Treats the *File* value as a file, even if a directory is specified. When a directory is specified with the *Directory* parameter, information about the directory itself is listed instead of information about the files it contains.

-D*Device* Specifies the name of the DOS device as **/dev/fd0** or **/dev/fd1**. The default device is **/dev/fd0**.

-e Uses the **-l** flag to write the list of clusters allocated to the file.

-l Produces a list of clusters that includes the creation date, size in bytes, and attributes of the file. The size of a subdirectory is specified as 0 bytes. The attributes have the following meanings:

A (Archive)
The file has not been backed up since it was last modified.

D (Directory)
The file is a subdirectory and not included in the normal DOS directory search.

H (Hidden)
The file is not included in the normal DOS directory search.

R (Read-only)
The file cannot be modified.

S (System)
The file is a system file and not included in the normal DOS directory search.

-t Lists the entire directory tree starting at the named directory.

-v Writes information to standard output about the format of the disk. Use this flag to verify that a device is a DOS disk.

Examples

To read a directory of the DOS files on **/dev/fd0**, enter:

```
dosdir
```

The command returns the names of the files and disk-space information.

```
PG3-25.TXT
PG4-25.TXT
PG5-25.TXT
PG6-25.TXT
Free space: 312320 bytes
```

To read a directory of the DOS files on **/dev/fd1**, enter:

```
dosdir -D/dev/fd1
```

The command returns the names of the files and disk-space information.

```
PG7-25.TXT
PG8-25.TXT
PG9-25.TXT
PG10-25.TXT
Free space: 312320 bytes
```

Files

/usr/bin/dosdir Contains the **dosdir** command.

Related Information

The **dosdel** command, **dosformat** command, **dosread** command, **doswrite** command.

Files in *Operating system and device management* describes files, file types, and how to name files.

---

dosformat Command

Purpose

Formats a DOS diskette.

Syntax

dosformat [**-V** *Label*] [**-D** *Device* | **-4**]

Description

The **dosformat** command formats a diskette with the DOS format.

The default device and DOS diskette drive format is **/dev/fd0** for a 3.5-inch diskette. The density is usually either 1.44M-byte or 2.88M-byte, depending on the density that the drive supports. Other DOS diskette drive formats are implemented by using the **-D** or **-4** flags.

To include a volume label, use the **-V** flag.

> **Note:** The purpose of this command is to facilitate file transfer between this operating system and DOS systems. Using this command to format a diskette that needs to have the DOS system startup files on it is not recommended.

Flags

-V Write the *Label* parameter to the diskette as the DOS volume label.

-D*Device* Specifies the diskette drive type and size. The *Device* parameter can be specified as:

For a 3.5-inch, 1.44M drive:

/dev/fd0
1.44MB (default)

/dev/fd0h
1.44MB

/dev/fd0l
720KB

/dev/fd0.18
1.44MB

/dev/fd0.9
720KB

For a 3.5-inch, 2.88M drive:

/dev/fd0
2.88MB (default)

/dev/fd0h
2.88MB

/dev/fdol
720KB

/dev/fd0.36
2.88MB

/dev/fd0.18
1.44MB

/dev/fd0.9
720KB

For a 5.25-inch, 1.2M drive:

/dev/fd0
1.2MB (default)

/dev/fd0.15
1.2MB

/dev/fd0.9
360KB

-4 Specifies the lower density for the diskette size.

Examples

1. To format a 3.5-inch, 1.44M-byte diskette with the volume label ″homework,″ type the following:
   ```
   dosformat -V homework
   ```
2. To format a 5.25-inch, 360K-byte diskette, type the following:
   ```
   dosformat -D /dev/fd1.9
   ```

   OR
   ```
   dosformat -D /dev/fd1 -4
   ```

Files

/usr/bin/dosformat	Contains the **dosformat** command.

Related Information

The **dosdel** command, **dosdir** command, **dosread** command, **doswrite** command.

---

dosread Command

Purpose

Copies DOS files.

Syntax

dosread [**-a**] [**-v**] [**-D** *Device*] *File1* [*File2*]

Description

The **dosread** command copies the DOS file specified by the *File1* variable to standard output or to the file specified by the *File2* variable. If no pathname is specified for the *File2* variable, the DOS file is copied to the root directory.

Unless otherwise specified, the **dosread** command copies the number of bytes specified in the directory entry for the file specified by the *File1* variable. This means, in particular, that you cannot copy directories because, by convention, directories have a record size of 0.

You can use DOS file-naming conventions with one exception: the \ (backslash). Because the \ character can have special meaning in DOS, use a / (slash) character as the delimiter to specify subdirectory names in a DOS path name. The **dosdir** command converts lowercase characters in the file or directory name to uppercase before it checks the disk. Because all file names are assumed to be full (not relative) path names, you need not add the initial / (slash).

Notes:

1. The **dosread** command does not interpret the * and ? (asterisk and question mark) wildcard characters as having special meaning. If you do not specify a file-name extension, the file name is matched as if you had specified a blank extension.
2. You cannot customize the name of this command. The command must be named **dosread**.
3. The **dosread** command reads files from the default drive containing the DOS diskette. The **dosread** command then copies the files to the current directory as a file recognized by this operating system. If the DOS diskette contains subdirectories, the **dosread** command does not create corresponding new subdirectories in this operating system. You must create the subdirectory and specify each DOS file you want to copy into the new subdirectory.

Flags

-a	Replaces each CR-LF (carriage return, line-feed) key sequence with a new-line character and interprets a Ctrl-Z (ASCII SUB) key sequence as the end-of-line character.
-D*Device*	Specifies the name of the DOS device as **/dev/fd0** or **/dev/fd1**. The default value of the *Device* variable is **/dev/fd0**. This device must have the DOS disk format.
-v	Writes file information to standard output about the format of the disk. Use this flag to verify that a device is a DOS disk.

Examples

1. To copy a text file from a DOS, type:

   ```
   dosread  -a chap1.doc chap1
   ```

   This command sequence copies the DOS text file `\CHAP1.DOC` on default device **/dev/fd0** to `chap1` in the current directory.
2. To copy a binary file from a DOS diskette, type:

   ```
   dosread -D/dev/fd1 /survey/test.dta /home/fran/testdata
   ```

   This command sequence copies the DOS data file `\SURVEY\TEST.DTA` on **/dev/fd1** to `/home/fran/testdata`.
3. To copy every DOS file on a diskette, type:

   ```
   dosdir | awk '!/There are/ {print $1}'|xargs -t -i dosread {} {}
   ```

   This command sequence takes files from the default drive containing the DOS disk and copies them to the current directory.

Files

/usr/bin/dosread	Contains the **dosread** command.
/dev/fd0	Contains the device name for a diskette drive.

Related Information

The **awk** command, **dosdel** command, **dosdir** command, **dosformat** command, **doswrite** command, **xargs** command.

"Directories" in *Operating system and device management*.

"Types of files" in *Operating system and device management* describes files, file types, and how to name files.

---

doswrite Command

Purpose

Copies files to DOS files.

Syntax

doswrite [**-a**] [**-v**] [**-D***Device*] *File1 File2*

Description

The **doswrite** command copies the file specified by the *File1* parameter to the DOS file specified by the *File2* parameter. The **doswrite** command copies files to a single DOS diskette. The **doswrite** command cannot copy files across multiple DOS diskettes.

The **doswrite** command writes the file specified by the *File2* parameter to the DOS device using standard DOS naming conventions. Because the DOS \ (backslash) character can have a special meaning for the DOS operating system, do not use a \ (backslash) when specifying subdirectory names in the *File2* parameter. Use the / (slash) character instead.

The **doswrite** command converts lowercase characters specified in the *File1* parameter to uppercase before it checks the DOS device. Because all file names are assumed to be full (not relative) path names, you do not need to add the initial / (slash).

If the file specified in the *File2* parameter contains a / (slash), each intervening component must exist as a directory and the last component (the named file) must not exist. Any existing file with the same name is overwritten.

Notes:

1. The wildcard characters * and ? (asterisk and question mark) are not treated in a special way by this command (although they are by the shell). If you do not specify a file-name extension, the file name is matched as if you had specified a blank extension.
2. This command must be named **doswrite**.
3. A DOS directory holds up to 244 files.

Flags

-a	Replaces NL (new-line) characters with the CR-LF (carriage return, line-feed) sequence. Ctrl-Z is added to the output at the end of file.
-D *Device*	Specifies the name of the DOS device as **/dev/fd0** or **/dev/fd1**. The default device is **/dev/fd0**. This device must have the DOS disk format.
-v	Writes information to standard output about the format of the disk. Use this flag to verify that a device is a DOS disk.

Examples

1. To copy a text file to a DOS diskette, enter:

   ```
   doswrite  -a chap1 chap1.doc
   ```

   This copies the file `chap1` in the current directory to the DOS text file `\CHAP1.DOC` on default device **/dev/fd0**.
2. To copy a binary file to a DOS diskette, enter:

   ```
   doswrite -D/dev/fd1 /home/fran/testdata /survey/test.dta
   ```

   This copies the data file `/home/fran/testdata` to the DOS file `\SURVEY\TEST.DTA` on **/dev/fd1**.
3. To copy every file in the current directory to a DOS diskette in your default drive, enter:

   ```
   for i in *
   do
   doswrite $i $i
   done
   ```

Files

/usr/bin/doswrite	Contains the **doswrite** command.
/dev/fd0	Contains the device name for diskette drive.

Related Information

Files in *Operating system and device management* describes files, file types, and how to name files.

The **dosdel** command, **dosdir** command, **dosformat** command, **dosread** command.

dp Command

Purpose
Parses and reformats dates.

Syntax
dp [**-form** *File* | **-format** *String*] [**-width** *Number*] *Date*

Description
The **dp** command parses and reformats dates. The **dp** command is not started by the user. The **dp** command is called by other programs, typically by its full path name, **/usr/lib/mh/dp**.

The **dp** command parses each mail header string specified as a date and attempts to reformat the string. The default output format for the **dp** command is the ARPA RFC 822 standard. For each string it is unable to parse, the **dp** command displays an error message.

Parameter
Date Specifies the date to be parsed.

Flags
-form *File* Reformats the date specified in the *Date* parameter to the alternate format described by the *File* variable.

-format *String* Reformats the date specified in the *Date* parameter to the alternate format specified by the *String* variable. The default format string follows:

```
%<(nodate{text})error:%{text}%|%(putstr(pretty{text}))%>
```

-help Lists the command syntax, available switches (toggles), and version information.

Note: For Message Handler (MH), the name of this flag must be fully spelled out.

-width *Number* Sets the maximum number of columns the **dp** command uses to display dates and error messages. The default is the width of the display.

Files
$HOME/.mh_profile Contains the MH user profile.

/etc/mh/mtstailor Contains MH command definitions.

Related Information
The **ap** command.

Mail applications in *Networks and communication management*.

dpid2 Daemon

Purpose
Starts the **dpid2** DPI-SMUX converter daemon as a background process.

Syntax

dpid2 [**-d** [*Level*]]

Description

The **dpid2** command starts the **dpid2** DPI-SMUX converter daemon. This command may only be issued by a user with root privileges or by a member of the system group.

The **dpid2** DPI-SMUX converter daemon complies with the standard Simple Network Management Protocol Distributed Protocol Interface Version 2.0 defined by RFC 1592 and SNMP MUX Protocol and MIB defined by RFC 1227.

dpid2 acts as a DPI 2.0 to SMUX converter. It is used to allow DPI sub-agents, such as **/usr/sbin/hostmibd**, to talk with the AIX SNMP version 1 agent. The converter changes DPI2 messages into SMUX protocol messages and vice-versa. **dpid2** itself is implemented as SMUX peer. It will connect with the TCP port 199 of the SMUX server which is part of **snmpd** agent. To a DPI2 sub-agent (e.g. **/usr/sbin/hostmibd**), **dpid2** behaves as a DPI2 agent. It listens on an arbitrary TCP port for a connection request from a DPI2 sub-agent. This is port number is registered by **dpid2** daemon with the **snmpd** agent through MIB variable dpiPortForTCP (1.3.6.1.4.1.2.2.1.1.1). The DPI2 sub-agent learns this port number from the **snmpd** agent by sending a get-request query for the dpiPortForTCP.0 (1.3.6.1.4.1.2.2.1.1.1.0) instance to the **snmpd** agent. After the DPI2 sub-agent knows the TCP port number which the DPI2 agent is listening on, it will then try to connect to it.

The **dpid2** daemon is normally executed during system startup when the **/etc/rc.tcpip** shell script is called.

The **dpid2** daemon should be controlled using the System Resource Controller (SRC). Entering **dpid2** at the command line is not recommended.

Use the following SRC commands to manipulate the **dpid2** daemon:

startsrc
: Starts a subsystem, group of subsystems, or a subserver.

stopsrc
: Stops a subsystem, group of subsystems, or a subserver.

refresh
: Causes a subsystem or group of subsystems to reread the appropriate configuration file.

lssrc Gets the status of a subsystem, group of subsystems, or a subserver.

Note: On AIX release 5.2.0 and after, the **snmpdv3** agent itself acts as a DPI2 agent and listens on the dpiPortForTCP.0 TCP port. Therefore, **dpid2** is not needed when using the **snmpdv3** agent. Therefore, the **dpid2** daemon won't be executed in the system startup and the **dpid2** line in **/etc/rc.tcpip** will be commented out.

Flags

-d *Level* Specifies tracing/debug level.

8	DPI level 1
16	DPI level 2
32	Internal level 1
64	Internal level 2
128	Internal level 3

Add the numbers for multiple trace levels.
Note: If the **-d** flag is specified, by the level number is not specified, the default level will be 56. If **-d** flag is not specified, the default level is 0.

Examples

1. To start the **dpid2** daemon, enter a command similar to the following:

   ```
   startsrc -s dpid2 -a "-f /tmp/dpid2.log"
   ```

   This command starts the **dpid2** daemon and logs information to the **/tmp/dpid2.log** file at debug level 0.
2. To stop the **dpid2** daemon, normally enter:

   ```
   stopsrc -s dpid2
   ```

   This command stops the **dpid2** daemon. The **-s** flag specified the subsystem that follows to be stopped.
3. To get the short status from the **hostmbid**, enter:

   ```
   lssrc -s dpid2
   ```

   This command returns the name of the daemon, the process ID of the daemon, and the state of the daemon (active or inactive).

Files

/etc/snmpd.conf	Specify smux peer entry in snmpd v1 agent configuration file.
/etc/snmpd.peers	Specify the configuration for smux peer.
/etc/mib.defs	Defines the Management Information Base (MIB) variables the SNMP agent and manager should recognize and handle.

Related Information

The **snmpdv1** daemon, **hostmibd** command.

The **/etc/snmpd.peers** file.

---

drm_admin Command

Purpose

Administers servers based on the Data Replication Manager (DRM), such as **glbd**, the replicated version of the global location broker (GLB).

Syntax

drm_admin [**-version**]

Description

The **drm_admin** tool administers servers based on the Data Replication Manager (DRM) such as **glbd**, the replicated version of the global location broker (GLB).

With **drm_admin**, you can inspect or modify replica lists, merge databases to force convergence among replicas, stop servers, and delete replicas.

The role of **drm_admin** is to administer the replication of databases, not to change the data they contain. For instance, you can use **drm_admin** to merge two replicas of the GLB database, but you must use **lb_admin** to add a new entry to the database. Also, although **drm_admin** can stop or delete a GLB replica, you must invoke **glbd** directly if you want to start or create a replica.

Once invoked, **drm_admin** enters an interactive mode, in which it accepts the commands described below.

Flags

-version Displays the version of NCS that this **glbd** belongs to, but does not start the daemon.

Subcommands

Most **drm_admin** commands operate on a default object (*DefaultObj*) at a default host (*DefaultHost*). Together, *DefaultObj* and *DefaultHost* specify a default replica. Defaults are established by the set command and are remembered until changed by another set.

Currently, the only known object is GLB.

Some **drm_admin** commands operate on a host other than the default. We identify this host as *OtherHost*.

The host name you supply as a *DefaultHost* or an *OtherHost* takes the form *Family:Host*, where the host can be specified either by its name or by its network address. For example, `ip:jeeves`, `ip:bertie`, and `ip:#192.5.5.5` are acceptable host names.

addrep *OtherHost*
Adds *OtherHost* to the replica list at *DefaultHost*. The replica at *DefaultHost* will propagate *OtherHost* to all other replica lists for *DefaultObj*.

chrep -from *OtherHost* **-to** *NewOtherHost*
Changes the network address for *OtherHost* in the replica list at *DefaultHost* to *NewOtherHost*. The replica at *DefaultHost* will propagate this change to all other replica lists for *DefaultObj*. The **chrep** command will fail if a replica of *DefaultObj* is running at *OtherHost* or if *OtherHost* is not on the replica list at *DefaultHost*.

delrep *OtherHost*

Deletes the replica of *DefaultObj* at *OtherHost*. The **delrep** command tells the replica at *OtherHost* to:

1. Propagate all of the entries in its propagation queue.
2. Propagate a delete request to all other replicas, causing *OtherHost* to be deleted from all other replica lists for *DefaultObj.*
3. Delete its copy of *DefaultObj.*
4. Stop running.

The **delrep** command returns you immediately to the **drm_admin** prompt, but the actual deletion of the replica can take a long time in configurations that are not stable and intact. You can check whether the daemon for the deleted replica has stopped by listing the processes running on its host.

info

Gets status information about the replica for *DefaultObj* at *DefaultHost*.

lrep [**-d**] [**-clocks**] [**-na**]

Lists replicas for *DefaultObj* as stored in the replica list at *DefaultHost*.

-d Lists deleted as well as existing replicas.

-clocks
Shows the current time on each host and indicates clock skew among the replicas.

-na Lists the network address of each host.

merge {**-from** | **-to**} *OtherHost*

Copies entries in the *DefaultObj* database and replica list from one replica to another. It copies an entry if no corresponding entry exists in the destination database or if the corresponding entry in the destination database bears an earlier timestamp.

A merge does not cause entries to be propagated. The database and replica list at the origination are not changed.

The **-from** option copies entries from the *DefaultObj* database and replica list at *OtherHost* to the *DefaultObj* database and replica list at *DefaultHost*.

The **-to** option copies entries from the database and replica list at *DefaultHost* to the database and replica list at *OtherHost*.

A **merge -from** followed by a **merge -to** causes the replicas at the two hosts to converge.

merge_all

Uses *DefaultHost* as the hub for a global merge of all replicas for *DefaultObj*. For each host on the replica list at *DefaultHost*, a **merge_all** first does a **merge -from**, then does a **merge -to**. All replicas of *DefaultObj* are thereby forced into a consistent state. The **merge_all** operation does not cause any entries to be propagated.

You should do a **merge_all** when:

A replica is purged.

A replica is reset.

A replica has been inaccessible for two weeks or more.

A replica has become physically inaccessible (for example, when its database is destroyed by a disk failure)

monitor [**-r n**]

This command causes **drm_admin** to read the clock of each replica of *DefaultObj* every n minutes and to report any clock skews or nonanswering replicas. If you do not specify **-r**, the period is 15 minutes.

purgerep *OtherHost*
Purges *OtherHost* from the replica list at *DefaultHost*. The replica at *DefaultHost* then propagates a delete request to the replicas at the hosts remaining on its list, thereby removing *OtherHost* from all other replica lists for *DefaultObj*. The delete request is not sent to *OtherHost*.

A **purgerep** can cause data to be lost and should only be used when a replica has become physically inaccessible. You should do a **merge_all** operation after the **purgerep** to prevent the remaining replicas of the *DefaultObj* database from becoming inconsistent. If the purged replica is still running, it should be reset.

We recommend that you use **chrep** (rather than **addrep** and **purgerep**) to change entries on the replica list.

quit
Quits the **drm_admin** session.

reset *OtherHost*
Resets the replica of *DefaultObj* at *OtherHost*.

The **reset** command tells the replica at *OtherHost* to delete its copy of *DefaultObj* and to stop running. It does not cause *OtherHost* to be deleted from any other replica lists. This command can cause data to be lost unless a successful **merge_all** is done first.

set [**-o** *ObjName*] **-h** *HostName*
Sets the default object and host. All subsequent commands will operate on *ObjName*. Subsequent commands that do not specify a host will be sent to *HostName*. If you do not specify the **-o** option, **drm_admin** keeps the current *DefaultObj*.

If you use set with the **-o** option, **drm_admin** checks the clocks at all hosts with replicas of the specified object.

stop
Stops the server for *DefaultObj* that is running at *DefaultHost*.

Example

The following example starts **drm_admin**, sets the default object to GLB, and sets the default host to `mars`:

```
/etc/ncs/drm_admin drm_admin: set -o glb -h dds:mars
 Default object: glb default host: dds:mars
 state: in service
 Checking clocks of glb replicas
 dds:mars 1987/04/09.17:09
 dds:pluto 1987/04/09.17:09
 dds:mercury  1987/04/09.17:07
```

Related Information

The **lb_admin** command.

The **glbd** (NCS) daemon

drmgr Command

Purpose

The **drmgr** command can be used to install and configure dynamic logical partitioning (DLPAR) scripts.

Syntax

drmgr { **-i***script_name* [**-w** *minutes*] [**-f**] | **-u** *script_name* } [**-D***hostname*]

drmgr [**-b**]

drmgr [**-R** *script_install_root_directory*]

drmgr [**-S** *syslog_ID*]

drmgr [**-l**]

Description

DLPAR scripts are provided by system administrators and vendors to coordinate the consumption of resources (for example, specific processors and large amounts of pinned memory) by applications and/or middleware with the addition or removal of those resources with respect to the operating system. DLPAR scripts are invoked both before and after DLPAR operations. DLPAR scripts are provided so that applications can be cleanly quiesed and restarted.

When installing scripts, the **drmgr** copies the script to a private repository. The default location, of which is **/usr/lib/dr/scripts/all**. The user may specify an alternate location for this repository through the **-R** *base_script_directory* option. In addition, a user may also install scripts to be executed only on selected host machines by specifying the **-D** *hostname* option. The *hostname* parameter serves as an extension to the base path and is compared to the current hostname using the '**uname -n**' command. If the **-D** parameter is used to install a script, then it has to be used to uninstall it.

Note that the various action flags specified above cannot be combined. That is, a user cannot combine **-R** and **-S** flags, **-l** and **-R** flags and so on.

Flags

-b	This option will rebuild the scripts information file managed by **drmgr**. In general, this option should only be used when restoring scripts from another systems.
-D *hostname*	This flag specifies the hostname of the machine on which the script can be invoked.
-f	Forces the replacement of an existing script.
-i *script_name*	This flag is used to install a script. The *script_name* is the script to be installed with complete path. If the path is not specified, the current directory is assumed. In case of any name conflicts, **drmgr** will issue a warning and will not install the script. Any existing script can be overwritten by specifying **-f** flag.
-l	This option will display the details regarding the DLPAR scripts that are currently installed.
-R *base_script_directory*	This option can be used to change the base script install directory.
-S *syslog_ID*	This ID string will be used as the syslog ID string while logging the syslog messages. Note that this ID string is appended to every entry logged in syslog by **drmgr**.
-u *script_name*	Uninstalls a DLPAR script. If the script was installed with the **-D** option, then the same parameter should be used to uninstall it. If no directory is specified, **drmgr** will try to remove the DLPAR script from ″all″ install directory.
-w *minutes*	Overrides the time limit value specified by the vendor for the script. The script will be aborted if it exceeds the specified time limit.

Exit Status

0 Successfully completed the requested operation

>0 The command failed. The cause of failure may be one of the following:

- File/Directory does not exist.
- The length of the parameter exceeds the system limit (PATH_MAX).
- Too many arguments were specified.
- You must have root authority to run this command.

Related Information

Dynamic logical partitioning in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

---

drslot Command

Purpose

Manages a dynamically reconfigurable slot, such as, a hot plug slot.

Syntax

To Identify a Hot Plug Slot

drslot -i { -s *Slot* | **-l** *DeviceName* } **-c** *ConnectorType*

To Prepare a Hot Plug Slot for Configuring Devices

drslot -a -s *slot* **-c** *ConnectorType* [**-l**]

To Prepare a Hot Plug Slot for Removal of a Device

drslot -r { **-s** *slot* | **-l** *DeviceName*} **-c** *ConnectorType* [**-l**]

To Prepare a Hot Plug Slot for Removal and Replacement of a Device

drslot -R { **-s** *slot* | **-l** *DeviceName* } **-c** *ConnectorType* [**-l**]

Description

The **drslot** command manages dynamically reconfigurable slots, that is, hot plug slots. Hot plug slots are the plug-in points for connecting entities which can be configured without turning the system power off or rebooting the operating system. For the add (**-a**) operation, the slot must be specified directly by using the **-s** flag, giving the unique identifier for the slot. For the identify (**-i**), the remove (**-r**), and the replace (**-R**) operations, the slot may be specified directly with the **-s** flag, or indirectly. The slot may be specified indirectly by using the **-l** flag giving the logical name for a device connected to the slot. The **drslot** command determines to which slot the specified device is connected and manages that slot.

Notes:

1. The remove and replace operations fail unless the device connected to the identified slot has been unconfigured. For more information on how to successfully unconfigure a device, see Managing Hot Plug Connectors in *Operating system and device management*.
2. After an add or replace operation, you must run the **cfgmgr** command in order to make the new device active and ready for use by the operating system.

Flags

Note: Do not use the **-a**, **-i**, **-r**, **-R** flags together.

-a	Prepares a hot plug slot for configuring the device(s) connected to it. The slot is first identified to you and you are prompted for confirmation of the slot. Next, you are prompted for confirmation that the device has been connected to the slot. Upon confirmation that the device has been connected, the slot is prepared and the device is made ready for configuration.
-c *ConnectorType*	Specifies the *ConnectorType* of the *Slot* on which you are operating. For example, the *ConnectorType* for a hot plug PCI slot is `pci`. This flag is must be specified with the **-a**, **-i**, **-r**, and **-R** flags.

-i Identifies a hot plug slot. The identification of the slot is hardware dependent. For example, if a slot has an LED associated with it, issuing the **drslot -i** command may cause the LED to flash.

-I Specifies that the identification step should be skipped when using the **-a** (add), **-r** (remove), and **-R** (replace) flags. This flag should only be used when you are sure you have already identified the proper slot.

-l *DeviceName* Specifies the *DeviceName*, which is the logical device name of the device connected to the slot to be managed. This flag must be used for the **-i** (identify), **-r** (remove) or **-R** (replace) flags if the **-s** flag is not used.

-r Prepares a hot plug slot for removal of a device that has been previously unconfigured with the **rmdev** command, or the SMIT, or Web-based System Manager equivalent. The slot is identified and you are prompted for confirmation of the slot. If a visual indicator is associated with the slot, it is turned off. Finally, the slot is prepared for device removal and you are prompted for confirmation that the device has been removed from the slot.

-R Prepares a hot plug slot for the removal of a device that has been previously unconfigured and the replacement with an identical device. The device must be unconfigured with the **rmdev** command, or the SMIT or Web-based System Manager equivalent. **drslot** identifies the slot and you are prompted for confirmation of the slot. Next, the slot is prepared for the replacement of the device. You are then prompted to confirm that the device has been replaced. Upon confirmation that the device has been replaced in the hot plug slot, the slot is prepared and the device is made ready for configuration.

-s *Slot* Specifies the *Slot* on which **drslot** should operate. This flag is required for the add (**-a**) operation. This flag must be used for the identify (**-i**), remove (**-r**) or replace (**-R**) operations if the the **-l** flag is not used. The format of *Slot* is Platform/*ConnectorType* dependent.

Examples

1. To identify a specific PCI hot plug slot, enter:

   ```
   drslot  -i  -c pci  -s U0.1-P1-I3
   ```

   In this example, there is an LED associated with this slot. The system may display a message similar to the following:

   ```
   The visual indicator of the specified PCI slot has been set to the identify state. Press
   Enter to continue or enter x to exit.
   ```

   The LED for the slot specified by `U0.1-P1-I3` flashes until the you press the Enter key.

2. To add a hot pluggable Ethernet adapter to a hot plug slot without confirmation of the slot, enter:

   ```
   drslot  -a  -I  -c pci  -s U0.1-P1-I3
   ```

   No confirmation prompt is given for identifying the slot. There will be a confirmation prompt displayed when it is time to put the new adapter into the slot, and a message similar to the following displays:

   ```
   The visual indicator for the specified PCI slot has been set to the action state. Insert the
   PCI card into the identified slot, connect any devices to be configured, and press Enter to
   continue. Enter x to exit.
   ```

   After connecting the adapter, press Enter, and the slot is prepared.

3. To identify a particular PCI slot before replacing the scsi card in it, enter the following:

   ```
   drslot  -R  -c pci  -s U0.2-P1-I3
   ```

   The system displays messages similar to the following:

   ```
   The visual indicator of the specified PCI slot has been set to the identify state. Press
   Enter to continue or enter x to exit.
   ```

   The LED for the PCI slot blinks to identify the slot. Pressing any key but the Enter key exits the command. Pressing Enter continues with this slot. If continuing, the LED for the PCI slot is changed to the action state and the system displays a message similar to the following:

```
The visual indicator for the specified PCI slot has been set to the action state. Replace
the PCI card in the identified slot, reconnect any devices to be configured, and press Enter
to continue. Enter x to exit. Exiting now leaves the PCI slot in the removed state.
```

Files

/usr/sbin/drslot

Related Information

The **lsslot** command, **rmdev** command, the **cfgmgr** command.

For information about Hot Plug Management and PCI Hot Plug Support for PCI Adapters, see PCI hot plug management in *Operating system and device management*.

---

dscreen Command

Purpose

Starts the Dynamic Screen utility.

Syntax

dscreen [**-i** *InfoFile*] [**-t** *TermType*]

Description

The **dscreen** command starts the Dynamic Screen utility, which allows a single physical terminal to be connected to several virtual sessions, or screens, at one time.

If no flags are specified, the **dscreen** command reads the description for the terminal specified in the **TERM** environment variable from the file specified in the **DSINFO** environment variable. If the **DSINFO** environment variable is not specified, the terminal description is read from the **/etc/dsinfo** file. A terminal description typically contains the following configuration information:

- Keys used with the Dynamic Screen utility and their function
- Number of pages of screen memory the terminal has available
- Code sequences that must be sent or received to access and use Dynamic Screen features

Flags

-i *InfoFile*	Specifies the file that contains alternate key mappings for use with the Dynamic Screen utility. This option is useful when the originally defined Dynamic Screen keys conflict with one of your applications. If this flag is not specified, terminal configuration information is read from the file specified in the **DSINFO** environment variable, if set. Otherwise, information is read from the **/etc/dsinfo** file.
-t *TermType*	Identifies the terminal description to be read from the file containing the key mappings. This option is useful when the desired terminal type does not match the setting of the **TERM** environment variable.

Examples

1. To start the Dynamic Screen utility using key mapping defaults, enter:

   ```
   dscreen
   ```

   This sets the **DSINFO** and **TERM** environment variables as designated in the default **/etc/dsinfo** file.

2. To start the Dynamic Screen utility and specify a file that contains alternate key mappings and also identifies a terminal description to be read from the file, enter:

   ```
   dscreen -i myfile -t myterm
   ```

   This uses information from a user-created **dsinfo-**type file named `myinfo` to handle unusual key mapping needs. The `myinfo` file also contains a terminal definition named `myterm`.
3. To start the Dynamic Screen utility and specify an alternate terminal setup, enter:

   ```
   dscreen -t wy60-wp
   ```

   This terminal definition (maintained in the **/etc/dsinfo** file) sets **dscreen** assigned key actions so they do not conflict with control key command sequences in the word processing application being used.

Files

/etc/dsinfo	Contains the terminal descriptions for the Dynamic Screen utility.

Related Information

Dynamic screen utility in *Networks and communication management* book.

---

dslpaccept Command

Purpose

Accept print queue requests for directory-enabled System V print systems.

Syntax

dslpaccept *PrintQueueName*

Description

The **dslpaccept** and **dslpreject** commands are used to set a print queue so that it will accept or reject print requests being queued for it. Unlike the **accept** and **reject** commands, the directory-enabled commands can control remote print systems, so long as they are directory-enabled. This is because they write directly to the print queue object on the directory server.

The user of this command must be directory-enabled and have permissions set for write, modify, search and read on the directory, in the directory context in which the user is administrator.

Parameters

PrintQueueName	The *PrintQueueName* parameter is the RDN of the print queue object. Multiple print queue names may be specified in a comma-separated list.

Exit Status

0	Indicates success.
1	Indicates invalid options.
2	Indicates that the specified print queue is unknown.
3	Indicates that this user does not have modify permissions.
4	Indicates that an invalid RDN was supplied.
5	Indicates that the value is already set.

6 Indicates that the command is unable to contact the directory service

7 Indicates any other error.

Examples

1. To set the print queue ″hpcolor″ to accept requests:

```
dslpaccept hpcolor
```

Related Information

The **dslpaccess** command, **dslpadmin** command, **dslpdisable** command, **dslpenable** command, **dslpprotocol** command, **dslreject** command, **dslpsearch** command, **lpstat** command.

---

dslpaccess Command

Purpose

Allow or deny non-directory enabled users and systems access to a print queue for a System V print subsystem.

Syntax

dslpaccess -q *QueueName* **-a** *AllowList* | **-d** *DenyList*

Description

The **dslpaccess** command either allows or denies users and systems access to a directory-enabled print queue. It is modeled on the **lpadmin** command's **-u** option.

Allow and deny lists consist of a comma-separated list of entries, each of which may specify a login ID, or a system name and login ID, as follows:

[[*LoginID*]|[*System*!*LoginID*]],[[*LoginID*]|[*System*!*Login-ID*]],...

LoginID or *System*, or both, can be set to the wildcard **all**, allowing or denying all appropriate entries. Use **all** with care. When the **all** entry is added to one list, all non-**all** entries are removed from the other list, for the appropriate value of *LoginID* or *System*. The default for *System* is the local host.

The user of this command must be directory-enabled and have permissions set for write, modify, search and read on the directory, in the directory context in which they are administrator.

Flags

-a *AllowList*	Specifies a list of users to add to the allow list. If present, these are deleted from the deny list. This option can not be used with the **-d** option.
-d *DenyList*	Specifies a list of users to add to the deny list. If present, these are deleted from the allow list. This option can not be used with the **-a** option.
-q *QueueName*	The queue-name parameter is the RDN of the print queue. If the print queue name does not exist in the directory context, the command fails.

Exit Status

0 Indicates success.

1 Indicates invalid options.

2 Indicates that the specified print queue is unknown.

3 Indicates that the user does not have appropriate access control permissions.

4 Indicates that an invalid RDN was supplied.

5 Indicates that the value is already set.

6 Indicates any other error.

Examples

1. The following grants user fredb access to print queue printq1 on host systemX:
   ```
   dslpaccess -q printq1 -a systemX!fredb
   ```
2. The following denies access to print queue printq1 to user tomt for all hosts:
   ```
   dslpaccess -q printq1 -d all!tomt
   ```

Related Information

The **dslpaccept** command, **dslpadmin** command, **dslpdisable** command, **dslpenable** command, **dslpprotocol** command, **dslreject** command, **dslpsearch** command, **lpstat** command.

---

dslpadmin Command

Purpose

Configure directory-enabled print service for a System V print subsystem.

Syntax

dslpadmin [[**-q** *PrintQueueName* [**-D** *QueueDescription*] [**-n** *LocalQueueName*] [**-o banner** | **nobanner**] [**-A mail** | **none**] [**-F** *FaultRecovery*] [[**-P** *PhysicalPrinterName*] [**-s** *NetworkEntityName*]]] [**-P** *PhysicalPrinterName* [**-T** *PrinterType*] [**-l** *Location*] [**-L** *PDLList*]] [**-q** *PrintQueueName* **-P** *PhysicalPrinterName* [**-l** *ContentType*] [[**-i** *InterfaceScript*] | [**-m** [**Standard** | **PS**]]] [**-o** *PrintOptions*]] [**-q** *PrintQueueName* [**-l** *ContentType*]]] [**-q** *PrintQueueName* **-s** *NetworkEntityName* [**-a** *PrintSystemDNSName* | *PrinterSystemAddress*] [**-t BSD** | **HPNP**]]

dslpadmin [**-q** *PrintQueueName* [**-u** *PhysicalPrinterName*] [**-U** *ObjectRDN*]]

dslpadmin [**-x** *PrintQueueName*] [**-X** *PhysicalPrinterName*] [**-r** *NetworkEntityName*]

dslpadmin [**-h**]

Description

The **dslpadmin** command is used to perform the following functions in order to configure a directory-enabled print service:

- Add print queues and physical printers to the system.
- Modify print queues and physical printers.
- Remove print queues and physical printers from the system.
- Add and delete network entity objects for networked printers.

The dslpadmin command provides directory-aware versions of the functionality supplied by **lpadmin** (which is not directory-aware), and continues to use the traditional ``flat file'' configuration system. Note that where both systems are in use, the printer subsystem employs information found in the directory first. It is the responsibility of the administrator to ensure that naming conflicts do not arise between the two configuration systems.

The directory-enabled commands use Relative Distinguished Names (RDNs), rather than Distinguished Names (DNs). For example, to create a directory-enabled queue with a DN of ″cn=test,ou=printq,ou=print,cn=aixdata″, only the RDN ″test″ is to be used for the *PrintQueueName*.

When configuring a print queue where the administrator is not on the system that is to host the print queue, the *InterfaceScript* parameter of **-i** and the *PrinterType* parameter of **-T** are not checked. This is because the remote system cannot be accessed in order to do the checks. It is therefore the administrator's responsibility to ensure that the specified *InterfaceScript* and *PrinterType* exist on the remote hosting system.

A command line can contain any combinations of the **-q** , **-P** and **-s** flags, or any combination of the **-x**, **-X** and **-r** flags, but only one of each flag. When multiple directory objects are simultaneously created or modified, appropriate links are set up between the three object types (printers, print queues and network entities).

Flags

-a *PrinterSystemDNSName* | *PrinterSystemAddress*
: Associates a DNS name or network address with the system. If the argument given can be interpreted as an IPv4 or IPv6 address, it is an address, if not it is assumed to be a DNS name. The -a flag causes the network entity object specified by **-s** to be modified, or else created if it does not already exist. The administrator should ensure that network entity objects are given unique names, so as to avoid modifying existing UNIX system objects instead of adding new print system objects. This flag requires the **-s** flag.

-A [**mail** | **none**]
: Instructs the print system to generate a mail message if a print request fails. The mail is sent to the owner of the physical printer, or to the root user of the system hosting the print queue, if the printer has no owner or the user has no mail address. The default is **none**. This flag requires the **-q** flag.

-D *QueueDescription*
: Defines a description comment for the print queue object specified with the **-q** flag. This description is displayed whenever a user asks for a full description of a print queue using the **lpstat** command. Strings containing whitespace should be double quoted. This flag requires the **-q** flag.

-F *FaultRecovery*
: Defines the print queue's fault recovery mode. This flag specifies the recovery to be used if the printer on a print queue fails while printing a print request. The value of *FaultRecovery* can be any of the following:

  continue
  : Continue printing on the top of the page where printing stopped. This requires a filter to wait for the fault to clear before automatically continuing.

  beginning
  : Start printing the request again from the beginning.

  wait
  : Disable printing on *PhysicalPrinterName* and wait for the administrator or a user to enable printing again.

    During the wait the administrator or the user who submitted the stopped print request can issue a change request that specifies where printing should resume. If no change request is made before printing is enabled, printing resumes at the top of the page where it stopped, if the filter allows; otherwise, the request is printed from the beginning.

  The default value of *FaultRecovery* is **beginning**. This flag requires the **-q** flag.

-h
: Displays a brief help screen.

-i *InterfaceScript*
: Pathname for the printer's *InterfaceScript* when accessed through the specified print queue. This flag is not valid if the **-P** flag has not been specified. The interface scripts are usually supplied by the user. This flag cannot be used when **-m** has also been specified. This flag requires both the **-q** and the **-P** flags.

-I *ContentType*[, *ContentType*, ...] Specifies the print queue's content types. Allows the print queue to handle print requests with the content types in the list. If the list contains more than one *ContentType*, the *ContentType* parameters must be separated by commas. See the **lpadmin** manual page for a full description of the format. This also requires the **-P** flag and the **-q** flag.

-l *Location* Defines the printer's location. This is a string identifying where a printer is physically located, for example ″Building X, Room 6″. It can be searched on by the **dslpsearch** command. Once set, this value can only be overwritten, not removed. This flag requires the **-P** flag.

-L *PDL*[, *PDL*, ...] Specifies the list of Page Description Languages (PDLs) supported by the printer. This is used to advertise any PDL the printer supports, and can be searched on, using the **dslpsearch** command. The **AUTOSW**, **PCL**, **PCLXL**, **POSTSCRIPT**, **TEXT**, **ESCP**, **PJL**, **SIMPLE**, and **OTHER** PDLs are supported. If the **-L** flag is used to modify an existing physical printer object, the list replaces the existing list. This flag requires the **-P** flag.

-m [**standard** | **PS**] Model interface program for the printer when accessed through the specified print queue. It selects the model interface script to be used by the print queue. When a physical printer object is being created, and neither the **-m** nor the **-i** flag has been specified, the default is **standard**. This flag cannot be used when **-i** has also been specified. This flag requires both the **-q** and the **-P** flags.

-n *LocalQueueName* Defines the local name of a print queue. This name normally only differs from the queue's RDN when the queue is on a non-directory-enabled host. It is used by incoming remote network connections to identify the print queue on the receiving system. The default value is the print queue's RDN. This flag requires the **-q** flag.

-o [**banner** | **nobanner**] Defines if a banner page will always be produced by this print queue. The default value, **banner**, forces a banner page to be printed for all print requests, whereas **nobanner** allows the user to submit a print job specifying that no banner page is to be printed. This flag requires the **-q** flag.

-o *PrintOption*=*Value*[, ...] Specifies values for print options. See the **lpadmin** documentation for a detailed description of the print options available with the **-o** flag. This flag requires both the **-q** and the **-P** flags.

-P *PhysicalPrinterName* Create or modify a physical printer object. The *PhysicalPrinterName* argument specifies the RDN of a printer object. If the object does not already exist, **dslpadmin** creates it.

-q *PrintQueueName* **dlspadmin** Creates or modifies a print queue object. The *PrintQueueName* argument specifies the RDN of a print queue object. When adding a new print queue, you must specify the **-s** and **-P** flags so the command knows the *NetworkEntityName* and *PhysicalPrinterName* for the print queue being added. If the print queue object does not exist, **dslpadmin** creates it.

A command line can contain any combinations of the **-q**, **-P** and **-s**flags , or any combination of the **-x**, **-X** and **-r** flags, but only one of each flag. When multiple directory objects are simultaneously created or modified, appropriate links are set up between the three object types (printers, print queues and network entities).

-r *NetworkEntityName* Delete the network entity system object. Care needs to be taken not to delete a non-printer system object. It is the responsibility of the administrator to ensure that the correct object is deleted.

-s *NetworkEntityName* Specifies the network entity system object that hosts the print queue. If **-a** is also given, the object is created or modified. The *NetworkEntityName* argument specifies the RDN of an object in the current directory context. The network entity object defines the network address that remote clients need to use to access the print queue.

-t [**BSD** | **HPNP**] Defines the print protocol used by this ″networked printer″ print queue. Retry and timeout values are set to their default values for a networked printer. To change these values, the **dslpprotocol** command should be used. Note that this flag should only be used for networked printers supporting the BSD or HPNP protocol. This flag requires the **-q** flag.

-T *PrinterType*[, *PrinterType*, ...]	List of printer types. It identifies the printer as being of one or more printer types, for example ″hplaserjet″. See the **lpadmin** manual page for details. This flag requires the **-P** flag.
-u *PhysicalPrinterName*	Unlinks the named physical printer from the print queue (specified with the **-q** flag) without deleting its object. This flag requires the **-q** flag.
-U *ObjectRDN*	Unlinks either the physical printer or the print queue object (specified by *ObjectRDN*) from the print queue (specified with the **-q** flag), without deleting its object. This flag requires the **-q** flag.
-x *PrintQueueName*	Delete a print queue object.
-X *PhysicalPrinterName*	Delete a physical printer object.

Exit Status

0 Indicates success

255 (or -1)
Indicates an error in configuration. Error messages are displayed to explain the error or failure.

Examples

The following examples illustrate use of the dslpadmin command, when the user is logged on to a directory-enabled UNIX system.

1. The following adds an HP LaserJet network printer that uses the BSD remote print protocol, with a print queue RDN of ″denlj5n″, and a physical printer RDN of ″denplj5n″. It gives the print queue a description of ″HP JetDirect (PostScript)″, the printer type ″PS-b″, and the model interface script as ″PS″. The printer has a network address of ″p_hplj.ibm.com″:

   ```
   dslpadmin -q denlj5n -P denplj5n -T PS-b -D "HP JetDirect (Postscript)" \
   -I PS -m PS -A mail -o nobanner  -s denslj5n -a p_hplj.ibm.com -t BSD
   ```

   The print system will allow print requests of content type PS for this print queue, and allow disabling of banner pages.
2. The following adds an HP LaserJet PostScript network printer, using the HPNP remote print protocol, with a print queue RDN of ″dehpnp″, and a physical printer RDN of ″dephpnp″. It gives the print queue a description of ″HPNP (PCL)″, the printer type ″hplaserjet″, and the model interface script as ″standard″. The printer has a network address of ″p_hplj.ibm.com″:

   ```
   dslpadmin -q dehpnp -P dephpnp -T hplaserjet -D "HPNP (PCL)" -I pcl \
   -m standard -A mail -s deshpnp -a p_hplj.ibm.com -t HPNP
   ```

   The print system will allow print requests of content type PCL for this print queue, and reject requests if no banner page is requested. If a printer fault occurs, the print system will mail the owner of the printer.
3. The following deletes an HP LaserJet PostScript printer:

   ```
   dslpadmin -x delj5n  -X deplj5n
   ```
4. The following deletes an HPNP printer:

   ```
   dslpadmin -x dehpnp  -X dephpnp -r deshpnp
   ```

Related Information

The **cancel** command, **dslpaccept** command, **dslpaccess** command, **dslpenable** command, **dslpprotocol** command, **dslpsearch** command, **lp** command, **lpadmin** command, **lpstat** command.

dslpdisable Command

Purpose

Disable print queue requests for a System V print subsystem.

Syntax

dslpdisable [**-r** *Reason*] *PrintQueueName*

Description

The **dslpenable** and **dslpdisable** commands are used to enable or disable a print queue from processing print requests that have been queued for it. Unlike the **enable** and **disable** commands, the directory-enabled commands can control remote print systems, so long as they are directory-enabled. This is because they write directly to the print queue object on the directory server.

Flags

-r *Reason*
Assign the reason for disabling the print queue. Strings containing whitespace should be double quoted.

Reason is a string that is displayed by the lpstat command. No default reason is set when one is not specified.

Parameters

PrintQueueName
The *PrintQueueName* parameter is the RDN of the print queue. This could be a list of print queues. If the print queue name does not exist in the directory context, the command fails.

Exit Status

0 Indicates success.

1 Indicates invalid options.

2 Indicates that the specified print queue is unknown.

3 Indicates that this user does not have modify permissions.

4 Indicates that an invalid RDN was supplied.

5 Indicates that the value is already set.

6 Indicates that the command is unable to contact the directory service

7 Indicates any other error.

Example

To disable print queue ″printer1″, specifying the reason ″routine maintenance″, enter the following:

```
dslpdisable -r "routine maintenance" printer1
```

Related Information

The **dslpaccept** command, **dslpaccess** command, **dslpadmin** command, **dslpenable** command, **dslpprotocol** command, **dslreject** command, **dslpsearch** command, **lpstat** command.

dslpenable Command

Purpose

Enable print queue requests for a System V print subsystem.

Syntax

dslpenable *PrintQueueName*

Description

The **dslpenable** and **dslpdisable** commands are used to enable or disable a print queue from processing print requests that have been queued for it. Unlike the **enable** and **disable** commands, the directory-enabled commands can control remote print systems, so long as they are directory-enabled. This is because they write directly to the print queue object on the directory server.

Parameters

PrintQueueName
: The *PrintQueueName* parameter is the RDN of the print queue. This could be a list of print queues. If the print queue name does not exist in the directory context, the command fails.

Subcommands

Exit Status

0 Indicates success.

1 Indicates invalid options.

2 Indicates that the specified print queue is unknown.

3 Indicates that this user does not have modify permissions.

4 Indicates that an invalid RDN was supplied.

5 Indicates that the value is already set.

6 Indicates that the command is unable to contact the directory service

7 Indicates any other error.

Examples

1. To enable print queue ″hpcolor″, enter the following:

```
dslpenable hpcolor
```

Related Information

The **dslpaccept** command, **dslpaccess** command, **dslpadmin** command, **dslpdisable** command, **dslpprotocol** command, **dslreject** command, **dslpsearch** command, **lpstat** command.

dslpprotocol Command

Purpose

Configure the remote print protocol of print queue for a System V print subsystem.

Syntax

dslpprotocol -t *RemoteProtocol* [**-T** *TimeOut*] [**-R** *Retry*] [**-r**] *PrintQueueName*

dslpprotocol -l [**-S**] *PrintQueueName*

Description

The **dslpprotocol** command is used to configure the ″remote print protocol″ that a remote print client can use when sending print requests to a print queue.

In directory-enabled printing, to print to a remote print queue, the client must first get the remote print protocol it can use. This is obtained from the print queue object in the directory. This can be one or both of BSD and HPNP. Where more than one protocol is configured for a print queue, the UNIX print system uses the first value it reads, so a queue will normally only have a single protocol configured.

The *PrintQueueName* parameter is the Relative Distinguished Name (RDN) of the print queue. If the value assigned to *PrintQueueName* does not exist, the command fails.

The user of this command must be directory-enabled and have permissions set for write, modify, search and read on the directory, in the directory context in which they are administrator.

Flags

-l	Print out a description of the remote print protocol parameters associated with the print queue.
-t *RemoteProtocol*	Specifies the remote print protocol that can be used when sending print requests to this print queue. The protocol type values supported are **bsd** and **hpnp**. The default value is **bsd**.
-T *TimeOut*	Set the network connection timeout value for the specified protocol, that is, the time a network connection should stay alive in an idle condition before disconnection. The value n can also be specified in order to disable timing out. The value 0 causes the connection to be dropped as soon as it becomes idle. The default value is 10 minutes, and there is no practical upper limit. See the **lpsystem** manual page for a full definition of the **-T** option.
-r	This option is used to remove a specified protocol from the print queue object. This option requires that the **-t** option also be specified.
-R *Retry*	Set the network connection retry time for the specified protocol, that is, the time in minutes to wait before trying to re-establish the network connection after a failure. The default value is 2 minutes. A value of 0 causes the connection to be retried immediately. Note that this value must be shorter than the timeout value specified using the **-T** option. The value n can also be specified in order to prevent dropped connections being retried when no work is available. There is no practical upper limit on the value. For ″networked printers″, the retry time should be set to 0. See the **lpsystem** manual page for a full definition of the **-R** option.
-S	Used with the **-l** option to display the print queue's protocol setup in a simple format.

Parameters

PrintQueueName	The *PrintQueueName* parameter is the Relative Distinguished Name (RDN) of the print queue. If the value assigned to *PrintQueueName* does not exist, the command fails.

Exit Status

0 Indicates success.

1 Indicates invalid options.

2 Indicates that the specified print queue is unknown.

3 Indicates that this user does not have modify permissions.

4 Indicates that an invalid RDN was supplied.

5 Indicates that the value is already set.

6 Indicates any other error.

Examples

1. To set print queue ″printq1″ to allow the BSD remote print protocol, enter the following:
   ```
   dslpprotocol -t BSD printq1
   ```
2. To remove the BSD protocol from print queue ″hpcolor″, enter the following:
   ```
   dslpprotocol -r -t BSD hpcolor
   ```

Related Information

The **dslpaccept** command, **dslpaccess** command, **dslpadmin** command, **dslpenable** command, **dslpdisable** command, **dslpreject** command, **dslpsearch** command, **lpsystem** command.

---

dslpreject Command

Purpose

Reject print queue requests for directory-enabled System V print systems.

Syntax

dslpreject [**-r** *Reason*] *PrintQueueName*

Description

The **dslpaccept** and **dslpreject** commands are used to set a print queue so that it will accept or reject print requests being queued for it. Unlike the **accept** and **reject** commands, the directory-enabled commands can control remote print systems, so long as they are directory-enabled. This is because they write directly to the print queue object on the directory server. Print requests that are already queued are not affected by the **dslpreject** command.

The user of this command must be directory-enabled and have permissions set for write, modify, search and read on the directory, in the directory context in which the user is administrator.

Flags

-r *Reason* — Assigns a reason for the rejection. Strings containing whitespace should be within double quotes. *Reason* is a string that is displayed by the **lpstat** command. No default reason is set when one is not specified.

Parameters

PrintQueueName — The *PrintQueueName* parameter is the RDN of the print queue object. Multiple print queue names may be specified in a comma-separated list.

Exit Status

0 Indicates success.

1 Indicates invalid options.

2 Indicates that the specified print queue is unknown.

3 Indicates that this user does not have modify permissions.

4 Indicates that an invalid RDN was supplied.

5 Indicates that the value is already set.

6 Indicates that the command is unable to contact the directory service

7 Indicates any other error.

Examples

1. To set a print queue to reject requests and specify the reason that there is no toner, enter the following:

```
dslpreject -r "no toner" printer1
```

Related Information

The **dslpaccept** command **dslpaccess** command, **dslpadmin** command, **dslpdisable** command, **dslpenable** command, **dslpprotocol** command, **dslpsearch** command, **lpstat** command.

---

dslpsearch Command

Purpose

Search directory for print system objects on a System V print subsystem.

Syntax

dslpsearch [**-q** [**-p**]] | [**-P**] [**-o** *SearchOptions*]

Description

The **dslpsearch** command allows users and administrators to search the directory for print system objects. For example, a user could search for any printer that can print color PostScript files. The main use of this command will be to search for print queues that match the search string.

The **dslpsearch**command returns the Distinguished Name (DN) of any objects that match the search string. However, the Relative Distinguished Name (RDN) is required for use in the other directory-enabled commands. For example, if the DN ″cn=testqueue,ou=printq,ou=print,cn=aixdata″ is returned by the **dslpsearch** command, only the RDN ″testqueue" is used to refer to the print queue.

Flags

-q Search for print queues that match the search options. The search is done on the physical printer objects but the print queues that service those printers are displayed. This is the default search type. The **-q** option cannot be specified with **-P**.

-p This option is used with the **-q** option, and causes a list of physical printers servicing the print queue also to be displayed.

-P Search for physical printers that match the search string. The **-P** option cannot be specified with **-q**.

-o *SearchOptions* Multiple search options may form a comma-separated list. Each option may be constructed from the following:

- one or more of the following Page Description Languages (PDLs): **AUTOSW**, **PCL**, **PCLXL**, **POSTSCRIPT**, **TEXT**, **ESCP**, **PJL**, **SIMPLE**, **OTHER**
- any of the following printer facilities: **COLOR**, **DUPLEX**, **TRAYS**, **FINISH**
- one or more physical printer locations, specified by `location=xxxxxxxx` or `location='aaaa bbbbb'`
- The string value defined by location= is searched on with wildcards placed at both ends of the string, so `location=Room1` would find any printer with ″Room1″ in its location, such as ″Building X, Room1, Bay6″. The string value can also have wildcards (*) embedded in it, for example `location="Building X*Bay6"`. Multiple location values are OR'd in the search.
- The following are valid command lines containing search strings:

```
dslpsearch -q -o PCL,ESCP,location=room2,COLOR

dslpsearch -q -p -o "PS, location='Building 1, Room1', DUPLEX"
```

Exit Status

0 Indicates success.

1 Indicates invalid options.

2 Indicates that the search on the directory tree failed.

3 Indicates invalid directory context.

4 Indicates the command is unable to contact the directory service.

Examples

1. The following command line searches for any print queues that match the search options:
   ```
   dslpsearch -q -o search-options
   ```
2. The following searches for any physical printers that match the search options:
   ```
   dslpsearch -P -o search-options
   ```

Related Information

The **dslpaccept** command, **dslpaccess** command, **dslpadmin** command, **dslpdisable** command, **dslpenable** command, **dslpprotocol** command, **dslreject** command, **lpstat** command.

---

dspcat Command

Purpose

Displays all or part of a message catalog.

Syntax

To Display Messages in a Catalog

dspcat *CatalogName* [*SetNumber* [*MessageNumber*]]

To Format Output for the gencat Command

dspcat -g *CatalogName* [*SetNumber*]

Description

The **dspcat** command displays a particular message, all the messages in a set, or all the messages in a catalog. The **dspcat** command directs the messages to standard output.

Note: The **dspcat** command looks for the catalog files under the **NLSPATH** if the **LC_FASTMSG** is set to False in C or POSIX locale environment.

LC__FASTMSG specifies that default messages are used for the C and POSIX locales and that **NLSPATH** is ignored when **LC__FASTMSG** is set to True.

The default value for **LC__FASTMSG** will be True in **/etc/environment**.

The *CatalogName* parameter specifies a message catalog. The *SetNumber* parameter specifies a set in the catalog specified by the *CatalogName* parameter. The *MessageNumber* parameter specifies a particular message in the set specified by the *SetNumber* parameter. If you include all three parameters, the **dspcat** command displays the specified message. If you do not include the *MessageNumber* parameter, the **dspcat** command displays all the messages in the set. If you specify a nonexistent value for the *SetNumber* or *MessageNumber* parameter, the **dspcat** command displays an error message and returns a nonzero return value. If you specify only the *CatalogName* parameter, the **dspcat** command displays all the messages in the catalog. You must include the *SetNumber* parameter if you include the *MessageNumber* parameter.

The **dspcat** command uses the **NLSPATH** environment variable and the **LC_MESSAGES** category to find the specified message catalog if you do not use / (slash) characters in the value of the *CatalogName* parameter.

Flags

-g Formats output to be used as input to the **gencat** command. The *MessageNumber* parameter is not valid when you use the **-g** flag.

Examples

To display message number 2 in set number 1 of the `test.cat` file, enter:

```
dspcat test.cat 1 2
```

Files

/usr/bin/dspcat	Contains the **dspcat** command.

Related Information

The **dspmsg** command, **gencat** command, **mkcatdefs** command, **runcat** command.

The **catclose** subroutine, **catgets** subroutine, **catopen** subroutine.

For more information about the Message Facility, see Message Facility in *AIX Version 6.1 National Language Support Guide and Reference*.

---

dspmsg Command

Purpose

Displays a selected message from a message catalog.

Syntax

dspmsg [**-s** *SetNumber*] *CatalogName MessageNumber* ['*DefaultMessage*' [*Arguments*]]

Description

The **dspmsg** command displays either the text of a particular message from a message catalog generated with the **gencat** command or, if the message cannot be retrieved, a default message supplied as a parameter to the command. The **dspmsg** command directs the message to standard output. This command is intended for use in shell scripts as a replacement for the **echo** command.

Note: The **dspmsg** command looks for the catalog files under the **NLSPATH** if the **LC_FASTMSG** is set to False in C or POSIX locale environment.

LC__FASTMSG specifies that default messages are used for the C and POSIX locales and that **NLSPATH** is ignored when **LC__FASTMSG** is set to True.

The default value for **LC__FASTMSG** will be True in **/etc/environment**.

The **NLSPATH** environment variable and the **LC_MESSAGES** category are used to find the specified message catalog if no / (slash) characters are used in the value of the *CatalogName* parameter. If the catalog named by the *CatalogName* parameter is not found or if the message named by the *MessageNumber* parameter (and optional *SetNumber* value) is not found, then the supplied *DefaultMessage* value is displayed. If a *DefaultMessage* value is not specified, a system-generated error message is displayed.

The **dspmsg** command allows up to ten string arguments to be substituted into the message if it contains the **%s**, %n$s, **%ld**, or %n$ld **printf** subroutine conversion specification. Missing arguments for conversion specifications result in a **dspmsg** error message. Normal **printf** subroutine control character escapes (for example, **\n**) are recognized.

The use of **printf** subroutine format strings is recommended in the catalog. This format provides for correct insertion of arguments even if the format strings in the message are in a different order than the default message. You must enclose the default message in single quotation marks if using the **%n$s** notation for message inserts.

Flags

-s *SetNumber* Specifies an optional set number. The default value for the *SetNumber* variable is 1.

Examples

To display set number 1, message number 2 of the `test.cat` catalog, enter:

```
dspmsg  -s  1  test.cat  2  'message  %s  not  found'  2
```

If the message is not found, `message 2 not found` is displayed.

Files

/usr/bin/dspmsg Contains the **dspmsg** command.

Related Information

The **dspcat** command, **gencat** command, **mkcatdefs** command, **runcat** command.

The **catclose** subroutine, **catgets** subroutine, **catopen** subroutine.

For more information about the Message Facility, see Message Facility in *AIX Version 6.1 National Language Support Guide and Reference*.

dtaction Command

Purpose

Invokes a CDE action with specified arguments.

Syntax

dtaction [**-contextDir** *context_dir*] [**-execHost** *host_name*] [**-termOpts** *terminal_arguments*] [**-user** *user_name*] *action_name* [*action_arg*] ...

Description

The **dtaction** command allows applications or shell scripts, which are otherwise not connected into the CDE development environment, to invoke action requests. The action called *action_name* is called with the *action_arg* provided on the command line. A single *action_name* is required; the user can provide any number of *action_args*. Interpretation of the *action_name* and *action_args* depends on the definition of the action in the action database. The action might be defined in one of the system action database files, or in one of the user's private action database files.

The *action_args* are absolute or relative path names of files. The **dtaction** command passes this list of files on to the specified action.

Error dialogs are posted when the following conditions are detected:

- Desktop environment could not be initialized
- Invalid user or password
- Unable to change ID to the requested user
- No action name specified

Flags

contextDir *context_dir*	Specifies a default directory context if the definition of *action_name* does not define a current working directory for command actions.
execHost *host_name*	Specifies an alternative execution host, *host_name*, for a command action. If the action is not a command action, the **dtaction** command ignores this option. The action is attempted on *host_name* instead of the hosts specified in the action's EXEC_HOST specification. An error is posted if it is not possible to invoke the specified action on any eligible host.
termOpts *terminal_arguments*	Specifies arguments intended for the terminal emulator that is provided for command actions that are not of type NO_STDIO. If there are white-space characters in the *terminal_arguments* string, that string must be quoted to protect it from the shell. These arguments are passed unchanged to the terminal emulator, so the user must ensure that the strings are reasonable. In particular, *terminal_arguments* does not allow the argument that specifies the command to be run in a terminal emulator window (that is, using **dtterm1** with the **-e** flag).

user *user_name*
Specifies a user name. If **dtaction** is not currently running as that user, a prompt dialog collects the specified user password or the root user password. After a valid password is entered, the **dtaction** command changes so that it is running as the requested user and then starts the requested action.

Parameters

action_name
Specifies the name of the action to be invoked.

action_arg
Specifies the absolute or relative file names of files.

Environment Variables

DTDATABASESEARCHPATH
A comma-separated list of directories (with optional host: prefix) that tells the action service where to find the action databases.

Exit Status

The following exit values are returned:

0
Successful completion.

>0
An error occurred.

Security

The **dtaction** command is an application enabled by PAM with service name **dtaction**. If the user name specified by **user** *user_name* option is different from the login user name, The **dtaction** command authenticates the user before invoking the specified action. It is capable of performing PAM authentication as well as traditional authentication.

To use PAM for authentication system-wide, establish root user permissions and modify the value of the *auth_type* attribute in the **usw** stanza of the **/etc/security/login.cfg** file to `PAM_AUTH`.

The authentication mechanisms used when PAM is enabled depend on the configuration for the login service in **/etc/pam.conf**. The **dtaction** command requires an **/etc/pam.conf** entry for the **auth** module type. The following configuration is recommended in **/etc/pam.conf** for the **dtaction** service:

```
dtaction        auth            required        /usr/lib/security/pam_aix
```

Examples

1. To invoke an action, enter:

   ```
   dtaction Xterm
   ```

   This launches X Windows terminal emulator (Xterm).
2. To invoke an action on a remote host, enter:

   ```
   dtaction -execHost hostname Xterm
   ```

   This executes Xterm on the specified remote host.
3. To invoke an action as a different user, enter:

   ```
   dtaction -user username Xterm
   ```

This executes Xterm as the specified user.

Location

/usr/dt/bin/dtaction

Standard Error

The **dtaction** command writes diagnostic error messages to standard error, which is redirected to **$HOME/.dt/errorlog**.

Files

/etc/pam.conf	Determines PAM authentication mechanisms.
/etc/security/login.cfg	Determines PAM authentication system-wide.

Related Information

The “dtlogin Command” on page 213, “dtsession Command” on page 240.

dtappintegrate Command

Purpose

The Common Desktop Environment application integration tool.

Syntax

dtappintegrate -s *ApplicationRoot* [**-t** *TargetPath*] [**-l** *Language*] [**-u**]

Description

The **dtappintegrate** command links the application CDE configuration files from application-specific locations to system locations and updates the system’s Browser help volumes for the languages affected. The **dtappintegrate** command is used during the installation process of an application. The application installation script should invoke the **dtappintegrate** command at the end.

There are four key subdirectories under the application root (referred to as **$APP_ROOT**) dictated by CDE policy. The directories are:

$APP_ROOT/dt/appconfig/types/ *Language*	For filetype, Front Panel, and action files.
$APP_ROOT/dt/appconfig/appmanager/ *Language*	For application group files.
$APP_ROOT/dt/appconfig/icons/ *Language*	For icons used by the CDE managers.
$APP_ROOT/dt/appconfig/help/ *Language*	For application help. For example, the default-language application **SpreadSheet** would load its desktop icons under: **/opt/SpreadSheet/dt/appconfig/icons/C/*.bm** and **/opt/SpreadSheet/dt/appconfig/icons/C/*.pm**, where **/opt/SpreadSheet** is the value of **$APP_ROOT**.

Note: $APP_ROOT is a syntactical convention of this document and is not used by the runtime environment.) All of these CDE configuration files and subdirectories are placed under a common top and should always include the default language subdirectory **C**.

In the simplest case, the command takes as input the application root, for example, **/opt/thisapp**. The outputs from this operation are corresponding subdirectories and files on the application server that contain relative symbolic links to the applications CDE configuration files described above, under the following system locations:

/etc/dt/appconfig Top-level application configuration subdirectory, consists of following subdirectories:

/etc/dt/appconfig/types/*Language***/**
Contains the ***.dt** and any ***.fp** links.

/etc/dt/appconfig/appmanager/*Language***/**
Contains links to the application group subdirectory and the action script files to appear as actions under the Application Manager.

/etc/dt/appconfig/help/*Language***/**
Contains symbolic links to the help files installed under the application's root.

/etc/dt/appconfig/icons/*Language***/**
Contains symbolic links to the CDE icons for the application.

Flags

-s *ApplicationRoot* Integrates the application located at *ApplicationRoot*. This flag is required.

-t *TargetPath* Links the application CDE configuration files from the application-specific location to *TargetPath* rather than to the system locations. This flag is optional.

If the **-t** flag is supplied, the files are linked under the specified subdirectory. For example, specifying **-t /etc/dt/private** would cause the application help files to be symbolically linked under **/etc/dt/private/help/***Language*. This flag is typically used only by system administrators who want to create separate applications and not by the application post-installation script. By default (with no **-t** specified), the application subdirectory root is global to the application host. All applications installed on the host will have their configuration files copied to the same place for merging with other application configuration files.

-l *Language* Specifies the language to integrate. Basically, this flag indicates the directories under which to find the application CDE configuration files. If this parameter is not specified, all languages will be integrated. This parameter is optional.

-u Integration of application is canceled. This flag is optional.

dtlogin Command

Purpose

Performs a CDE login service.

Syntax

dtlogin [-**config** *configuration_file*] [-**daemon**] [-**debug** *debug_level*] [-**error** *error_log_file*] [-**nodaemon**] [-**resources** *resource_file*] [-**server** *server_entry*] [-**session** *session_program*] [-**udpPort** *port_number*]

Description

The **dtlogin** command supports the following key tasks:

- Launching **dtgreet** login screen for explicitly managed local and remote displays and XDMCP-managed remote displays.
- Accessing traditional terminal (character) login from GUI login screen
- Authenticating and logging in system-dependent users
- Launching the selected session

The **dtlogin** command provides services similar to those provided by **init**, **getty**, and **login** on character terminals, which include prompting for login and password, authenticating the user, and running a session. A *session* is defined by the lifetime of a particular process. In the traditional character-based terminal world, a session is the user's login shell process; in the DT context, it is the DT Session Manager. If the DT Session Manager is not used, the typical substitute is either a window manager with an exit option, or a terminal emulator running a shell, where the lifetime of the terminal emulator is the lifetime of the shell process that it is running. This reduces the X session to an emulation of the character-based terminal session. When the session is terminated, **dtlogin** resets the X server and (optionally) restarts the whole process.

The **dtlogin** command supports management of remote displays using the X Display Manager Control Protocol, Version 1.0. (XDMCP). When **dtlogin** receives an indirect query from XDMCP, it can run a chooser process to perform an XDMCP BroadcastQuery (or an XDMCP Query to specified hosts) on behalf of the display and offer a menu of possible hosts that offer XDMCP display management. This feature is useful with X terminals that do not offer a host menu.

Because **dtlogin** provides the first interface that users see, it is designed to be simple to use and easy to customize according to the needs of a particular site.

Login Window

The Login window allows users to enter a user ID and password, select a startup session, and select a startup locale. Users can also reset the X server or temporarily suspend the X server to access the character login prompt.

The contents of the Login window are as follows:

login field
Provides an entry field in which users enter their IDs.

password field
Provides an entry field in which users enter their passwords (no-echo).

OK button
Authenticates a user and launches a session.

Clear button
Clears login and password fields.

Options
Lets users select a locale name and login session type. It also lets users restart the X server or switch to a character login prompt (for local displays). The contents of the Options menu are as follows:

Languages
Displays the Languages menu. Selecting the language from the login screen Options menu immediately localizes the login screen and sets the **LANG** variable for the next session. Login screen localization and **LANG** return to the default value upon conclusion of the session. The contents of this menu can vary depending upon the locales installed on the system. They can be overridden by using the **languageList** resource. The default locale of C can be overridden using the language resource. The system or **languageList** locales specified are displayed as menu items in the Languages menu. Alternate text to be displayed can be specified for a given locale name by using the **languageName** resource.

No-windows
Displays character login prompt (local displays only).

Reload Login
Restarts the X Server and returns to login screen.

Resources
Lists resources to be used.

Sessions
Displays Sessions menu. Allows users to select which session type should be started upon login. Menu items include the following:

DT Session
Starts a regular desktop session (Xsession).

Fail-safe Session
Starts a fail-safe session (Xfailsafe).

Help Displays help messages.

Controlling the Server

The **dtlogin** command controls local servers using POSIX signals. The SIGHUP signal is expected to reset the server, closing all client connections and performing other clean up duties. The SIGTERM signal is expected to terminate the server. If these signals do not perform the expected actions, the **resetSignal** and **termSignal** resources can specify alternate signals.

To control remote servers that are not using XDMCP, **dtlogin** searches the window hierarchy on the display and uses the KillClient X protocol request in an attempt to clean up the terminal for the next session. This might not actually kill all of the clients, because only those that have created windows are noticed. XDMCP provides a more sure mechanism; when **dtlogin** closes its initial connection, the session is over and the terminal is required to close all other connections.

Controlling dtlogin

The **dtlogin** command responds to two signals: SIGHUP and SIGTERM. When it is sent a SIGHUP, **dtlogin** rereads the configuration file and the file specified by the servers resource, and determines whether entries have been added or removed. If a new entry has been added, **dtlogin** starts a session on the associated display. Entries that have been removed are disabled immediately, meaning that any session in progress is terminated without notice, and no new session is started. When sent a SIGTERM, **dtlogin** terminates all sessions in progress and exits. This can be used when shutting down the system.

Internationalization

All labels and messages are localizable. The **dtlogin.cat** message catalog contains the localized representations of the default labels and messages. The **dtlogin** command reads the appropriate message catalog indicated by the **LANG** environment variable and displays the localized strings. An option on the authentication screen allows the user to override the default language for the subsequent session. If the authentication screen has been localized for the selected language, the screen is redisplayed in that language; otherwise, it is displayed in the default language. In either case, the **LANG** environment variable is set appropriately for the resulting session.

The resource language is available in the dtlogin configuration file to change the default language for a display. The **languageList** resource is available in the **dtlogin** configuration file to override the default set of languages displayed on the authentication screen. The **languageName** resource is available to provide a mapping from locale names to the text displayed on the Language menu.

Authentication and Auditing

The **dtlogin** command is a login service enabled by PAM with service name **dtlogin**. The **dtlogin** client supports PAM authentication in addition to traditional local UNIX login and auditing. Additional authentication or auditing functions, such as Kerberos or B1 can be added by individual vendors.

To use PAM for system-wide authentication, establish root user permissions and modify the value of the *auth_type* attribute in the **usw** stanza of the **/etc/security/login.cfg** file to `PAM_AUTH`.

The authentication mechanisms used when PAM is enabled depend on the configuration for the login service in **/etc/pam.conf**. The **dtlogin** command requires an **/etc/pam.conf** entry for the **auth**, **account**, **password**, and **session** module types. The following configuration is recommended in **/etc/pam.conf** for the **dtlogin** service:

```
dtlogin         auth              required        /usr/lib/security/pam_aix
dtlogin         account           required        /usr/lib/security/pam_aix
dtlogin         password          required        /usr/lib/security/pam_aix
dtlogin         session           required        /usr/lib/security/pam_aix
```

X Server Security

The X server provides both user-based and host-based access control. By default, **dtlogin** uses user-based access control to the X server (MIT-MAGIC-COOKIE-1). This level of security allows access control on a per-user basis. It is based on a scheme where if a client passes authorization data that matches what the server has, the client is allowed access. When a user logs in, this authorization data is by default stored and protected in the **$HOME/.Xauthority** file.

However, using host-based access control mechanisms might be preferable in environments with unsecure networks, because user-based access control allows any host to connect if the host has discovered the private key. Another drawback to user-based access control is that R2 or R3 clients are unable to connect to the server.

The **authorize** resource controls whether user-based or host-based access control is used by **dtlogin**. See the xhost, and xauth commands for more information.

Resources

The **dtlogin** command is controlled by the contents of the **dtlogin** configuration file, which defaults to **/usr/dt/config/Xconfig**. Some resources control the behavior of **dtlogin** in general, and others can be specified for a particular display.

General Resources

The following **dtlogin** general resources are not display-specific and apply to all displays where appropriate.

accessFile

Class: AccessFile

ClassType:
String

Default:
Null

Description:
To prevent unauthorized XDMCP service and to allow forwarding of XDMCP IndirectQuery requests, this file contains a database of host names that are either allowed direct access to this machine or have a list of hosts to which queries should be forwarded to. Refer to the Xaccess file section for a description of the format. If this resource is not set, all hosts will be allowed XDMCP service.

authDir

Class: AuthDir

ClassType:
String

Default:
/var/dt

Description:
The directory name that **dtlogin** uses to temporarily store authorization files for displays using XDMCP.

autoRescan

Class: AutoRescan

ClassType:
Boolean

Default:
True

Description:
Controls whether **dtlogin** rescans the configuration file and server file after a session terminates and the files have changed. You can force **dtlogin** to reread these files by sending a SIGHUP signal to the main process.

daemonMode

Class: DaemonMode

ClassType:
Boolean

Default:
False

Description:
The **dtlogin** command can make itself into an unassociated daemon process. This is accomplished by forking and leaving the parent process to exit, then closing file descriptors and releasing the controlling terminal. This is inconvenient when attempting to debug **dtlogin**. Setting this resource to False disables **daemonMode**.

debugLevel

Class: DebugLevel

ClassType:
Int

Default:
0

Description:
A nonzero value specified for this integer resource enables debugging information to be printed. It also disables daemon mode, which redirects the information into the normally unuseful bit-bucket.

errorLogFile

Class: ErrorLogFile

ClassType:
String

Default:
NULL

Description:
Error output is normally directed at the system console. To redirect it, set this resource to any file name. This file contains any output directed to **stderr** by **Xsetup**, **Xstartup**, and **Xreset**.

errorLogSize

Class: errorLogSize

ClassType:
Int

Default:
50

Description:
This resource specifies the maximum size of the error log file in kilobytes. When the limit is reached, **dtlogin** deletes the oldest entries in the file until the file size is reduced to 75 percent of the maximum. After the file is truncated, any user who is accessing the error log file (for example, using cat or tail) will need to close the file and reopen it for access in order to see subsequent information that is logged to the file.

exportList

Class: ExportList

ClassType:
String

Default:
NULL

Description:
Contain a set of variable names separated by a space or tab. Each variable named is obtained from the **dtlogin** environment and loaded into the environment of the server and session. See the Environment section for details.

fontPathHead

Class: FontPathHead

ClassType:
String

Default:
NULL

Description:
Value that is prepended to the default X server font path.

fontPathTail

Class: fontPathTail

ClassType:
String

Default:
NULL

Description:
Value that is appended to the default X server font path.

keyFile

Class: KeyFile

ClassType:
String

Default:
/usr/dt/config/Xkeys

Description:
XDM-AUTHENTICATION-1 style XDMCP authentication requires that a private key be shared between **dtlogin** and the terminal. This resource specifies the file containing those values. Each entry in the file consists of a display name and the shared key. By default, **dtlogin** does not include support for XDM-AUTHENTICATION-1 because it requires DES, which is not generally distributable.

lockPidFile

Class: LockPidFile

ClassType:
Boolean

Default:
True

Description:
Controls whether **dtlogin** uses file locking to prevent multiple instances of **dtlogin** from executing concurrently.

networkDevice

Class: NetworkDevice

ClassType:
String

Default:
/dev/dtremote

Description:
For remote connections, the value for `line` in **/etc/utmp** must also exist as a device in the **/dev** directory for commands such as **finger** to operate properly. This resource specifies the path name of the **/dev** file **dtlogin** creates when a remote display connects. For most platforms, the file is created as a symbolic link to **/dev/null**. The specified value must start with `/dev/`, or else the value is discarded and no file is created.

pidFile

Class: PidFile

ClassType:
STring

Default:
NULL

Description:
The filename specified is created to contain an ASCII representation of the process-ID of the main **dtlogin** process. This can be used when sending signals to **dtlogin**. The **dtlogin** client also uses file locking to attempt to prevent more than one dtlogin from running on the same machine. See the **lockPidFile** resource for more information.

removeDomainname

Class: RemoveDomainname

ClassType:
Boolean

Default:
True

Description:
When computing the display name for XDMCP clients, **dtlogin** typically creates a fully qualified host name for the terminal. Because this is sometimes confusing, **dtlogin** removes the domain name portion of the host name if it is the same as the domain name for the local host when this variable is set.

requestPort

Class: RequestPort

ClassType:
int

Default:
177

Description:
Indicates the UDP port number that **dtlogin** uses to listen for incoming XDMCP requests. Unless the system needs to be debugged the system, the default value for this resource should remain.

servers

Class: Servers

ClassType:
String

Default:
:0 Local local /system_dependent_path/X :0

Description:
Either specifies a file name full of server entries, one per line (if the value starts with a slash), or a single server entry. Each entry indicates a display that should be managed constantly and that is not using XDMCP. The general syntax for each entry is as follows:

DisplayName DisplayClass DisplayType[@ite] [Command [options]]

where:

DisplayName
A value that can be passed in the **-display** option to any X program. This string is used in the display-specific resources to specify the particular display, so caution must be taken to match the names. For example, use `:0 local /usr/bin/X11/X :0` instead of `localhost:0 local /usr/bin/X11/X :0` if your other resources are specified as `Dtlogin._0.session`). A asterisk (*) in this field expands to **`hostname:0`** by **dtlogin**.

DisplayClass
The display class portion is also used in the display-specific resources as the class portion of the resource. This is useful if you have a large collection of similar displays (a group of X terminals, for example) and want to set resources for groups of them. When using XDMCP, the display is required to specify the display class. Refer to your X terminal documentation for information on a reasonably standard display class string for your device.

DisplayType
If specified as `local`, indicates that an X server should be started for this entry. A value of `remote` indicates that an existing X server should be attached.

@ite On local bitmaps, the user can choose a **Command Line Login** option using the login screen, which temporarily suspends the X-server and presents the traditional character `login:` prompt. The user can then log in and perform non-X related tasks. When the user finishes and logs out, the X-server is restarted, and the login screen is redisplayed. In order to support **Command Line Login** mode, the display must have an associated Internal Terminal Emulator (ITE) device. By default, **dtlogin** associates the ITE device ″console″ (**/dev/console**) with display `:0`. If your configuration does not match this default, specify `@device` for any displays with an associated ITE, and specify `@none` for all other displays listed in the servers file.

Command [options]
The string that starts the X server. The **dtlogin** client will always connect to the X server using the *DisplayName* specified, so you might need to specify an explicit connection number as an option to your X server (`:0` in the preceding example).

sysParmsFile

Class: SysParmsFile

ClassType:
String

Default:
/system_dependent_path

Description:
Specifies a file containing shell commands, one of which sets the time zone environment variable (**TZ**) for the system. If the time zone is set using the shell syntax `TZ=`, **dtlogin** can use this information to set the time zone for the user session.

timeZone

Class: TimeZone

ClassType:
String

Default:
NULL

Description:
Specifies the local time zone for **dtlogin**. It is loaded into the environment of **dtlogin** as the value of the **TZ** variable and inherited by all subsequent sessions. Some systems maintain a configuration file that contains the time zone setting (for example, **/etc/src.sh**). See also the **sysParmsFile** resource.

wakeupInterval

Class: WakeupInterval

ClassType:
Int

Default:
10

Description:
If the user selects **Command Line Login** mode from the login screen, **dtlogin** terminates the X-server and allows the traditional character-based login prompt `login:` to become visible. If the user does not log in within 2 times the **wakeupInterval** seconds, the X-server is restarted. After the user has logged in, **dtlogin** checks every **wakeupInterval** seconds to see if the user has logged out. If so, the X-server is restarted and the login screen is redisplayed.

Display Resources

The **dtlogin** command display resources can be specified for all displays or for a particular display. To specify a particular display, the display name is inserted into the resource name between `Dtlogin` and the final resource name segment. For example, `Dtlogin.expo_0.startup` is the name of the resource defining the startup shell file on the `expo:0` display. The resource manager separates the name of the resource from its value with colons, and separates resource name parts with dots, so **dtlogin** uses underscores (_) for the dots (`.`) and colons (`:`) when generating the resource name.

Resources can also be specified for a class of displays by inserting the class name instead of a display name. A display that is not managed by XDMCP can have its class affiliation specified in the file referenced by the servers resource. A display using XDMCP supplies its class affiliation as part of the XDMCP packet.

The following **dtlogin** general resources are not display-specific and apply to all displays where appropriate.

authorize

ClassClass:
Authorize

Type: Boolean

Default:
False

Description:
Authorize is a Boolean resource that controls whether **dtlogin** generates and uses authorization for the server connections. Refer also to the **authName** resource.

authName

ClassClass:
AuthName

Type: String

Default:
MIT-MAGIC-COOKIE-1

Description:
If the **authorize** resource is used, **authName** specifies the type of authorization to be used. Currently, **dtlogin** supports only MIT-MAGIC-COOKIE-1 authorization. XDM-AUTHORIZATION-1 could be supported, but DES is not generally distributable. XDMCP connections state which authorization types are supported dynamically, so **authName** is ignored in this case. Refer also to the **authorize** resource.)

authFile

ClassClass:
AuthFile

Type: String

Default:
NULL

Description:
Communicates the authorization data from **dtlogin** to the server, using the **-auth** server command line option. Keep this resource in a write-protected directory to prevent its erasure, which would disable the authorization mechanism in the server. If NULL, **dtlogin** generates a file name.

chooser

ClassClass:
Chooser

Type:

Default:

Description:
Specifies the program run to offer a host menu for indirect queries redirected to the special host name `CHOOSER`. The default is **/usr/dt/bin/dtchooser**. See the Xaccess file section.

cpp

ClassClass:
Cpp

Type: String

Default:
system dep.

Description:
Specifies the path of the C preprocessor that is used by **xrdb**.

environment

ClassClass:
Environment

Type: String

Default:
system dep.

Description:
Contains a set of *name*=*value* pairs separated by a space or tab. Each item is loaded into the environment of the server and session. See the Environment section for more information.

failsafeClient

ClassClass:
FailsafeClient

Type: String

Default:
/system_dep./xterm

Description:
If the default session fails to execute, **dtlogin** falls back to this program. This program is executed with no arguments, but executes using the same environment variables as the session would have had.

grabServer

ClassClass:
GrabServer

Type: Boolean

Default:
True

Description:
To improve security, **dtlogin** grabs the server and keyboard while reading the name and password. The **grabServer** resource specifies if the server should be held while the name and password is read. When FALSE, the server is ungrabbed after the keyboard grab succeeds; otherwise, the server is grabbed until just before the session begins.

grabTimeout

ClassClass:
GrabTimeout

Type: Int

Default:
3 seconds

Description:
Specifies the maximum time **dtlogin** will wait for the grab to succeed. The grab can fail if another client has the server grabbed, or possibly if the network latencies are very high. The **grabTimeout** resource has a default of 3 seconds; use this resource with care, because a user can be deceived by a look-alike window on the display. If the grab fails, **dtlogin** kills and restarts the server (if possible) and session. Some X-terminals cannot display their login screens while the server is grabbed. Setting **grabServer** to FALSE allows the screen to be displayed but opens the possibility that a user's login name can be stolen by copying the contents of the login screen. Because the keyboard is still grabbed and the password is not echoed, the password cannot be stolen.

language

ClassClass:
Language

Type: String

Default:
system dep.

Description:
Specifies the default setting for the **LANG** environment variable. If the **dtlogin** screen is localized for that language, it is displayed appropriately; otherwise, it is displayed in the C language The user can temporarily override this setting using an option on the login screen. When the subsequent session terminates, the **LANG** variable reverts to this setting.

languageList

ClassClass:
LanguageList

Type: String

Default:
NULL

Description:
Allows the user to override the default set of languages displayed in the Language menu of the login screen. It is useful if the set of languages actually used on a particular display is smaller than the set installed on the system. The resource value is a list of valid values for the **LANG** environment variable. Language values should be separated by one or more spaces or tabs.

languageName

ClassClass:
LanguageName

Type: String

Default:
NULL

Description:
Allows the user to override the default locale name displayed in the Language menu of the login screen with alternate text. This way, instead of users seeing a `En_US` item, they could see an `English (United States)` item instead. This resource is specified as **Dtlogin** *****_local_name_. **languageName**: _text_ as follows:

```
Dtlogin*En_US.languageName: English (United States)
Dtlogin*Fr_CA.languageName: French (Canadian)
```

openDelay

ClassClass:
OpenDelay

Type: Int

Default:
5 seconds

Description:
Specifies the duration (in seconds) between successive attempts to open reluctant servers.

openRepeat

ClassClass:
OpenRepeat

Type: Int

Default:
5 seconds

Description:
Specifies the number of successive attempts to open reluctant servers.

openTimeout

ClassClass:
OpenTimeout

Type: Int

Default:
30 seconds

Description:
Specifies the amount of time to wait while actually attempting to open reluctant servers. This time is the same as the maximum time spent in the **connect** system call.

pingInterval

ClassClass:
PingInterval

Type: Int

Default:
5 minutes

Description:
To discover when remote displays disappear, **dtlogin** occasionally pings them, using an X connection and sending XSync requests. The **pingInterval** resource specifies the time (in minutes) between successive ping attempts.

pingTimeout

ClassClass:
PingTimeout

Type: int

Default:
5 minutes

Description:
Specifies the maximum wait time (in minutes) for the terminal to respond to the request. If the terminal does not respond, the session is terminated. The **dtlogin** client does not ping local displays. A local session should never be terminated as a result of the server waiting (for remote file system service, for example) and not responding to the ping.

reset

ClassClass:
Reset

Type: String

Default:
NULL

Description:
specifies a program that is run (as root) after the session terminates. If this resource is not set, no program is run. The conventional name is **Xreset**. See the Xreset File.

resetForAuth

ClassClass:
ResetForAuth

Type: Boolean

Default:
False

Description:
During the original implementation of authorization in the sample server, the authorization file was reread at server reset time instead of when checking the initial connection. Because **dtlogin** generates the authorization information just before connecting to the display, an old server does not get current authorization information. This resource causes **dtlogin** to send SIGHUP to the server after setting up the file, causing an additional server reset to occur, during which time the new authorization information is read.

resetSignal

ClassClass:
Signal

Type: Int

Default:
1 SIGHUP

Description:
Specifies the signal **dtlogin** sends to reset the server.

resources

ClassClass:
Resource

Type: String

Default:
NULL

Description:
Specifies the name of the file to be loaded by **xrdb** as the resource database onto the root window of screen 0 of the display. This resource database is loaded just before the authentication procedure is started, so it can control the appearance of the login window. See the section on the authentication screen, which describes the various resources that are appropriate to place in this file. There is no default value for this resource, but the conventional name is **Xresources**.

session

ClassClass:
Session

Type: String

Default:
/usr/dt/bin/Xsession

Description:
Specifies the session to be executed for the authenticated user. By default, the **/usr/dt/bin/Xsession** file is run. The conventional name is **Xsession**. Refer to the Xsession file.

setup

ClassClass:
Setup

Type: String

Default:
NULL

Description:
Specifies a program that is run (as root) prior to the display of the authentication screen. By default, no program is run. The conventional name is **Xsetup**. Refer to the Xsetup file.

startAttempts

ClassClass:
StartAttempts

Type: Int

Default:
4

Description:
Four numeric resources control the behavior of dtlogin when attempting to open reluctant servers: **openDelay**, **openRepeat**, **openTimeout**, and **startAttempts**. This resource specifies the number of times the entire process occurs before giving up on the server. After **openRepeat** attempts have been made, or if **openTimeout** seconds elapse in any particular attempt, **dtlogin** terminates and restarts the server, attempting to connect again. This process is repeated **startAttempts** time, at which point the display is declared dead and disabled.

startup

ClassClass:
Startup

Type: String

Default:
NULL

Description:
Specifies a program that is run (as root) after the authentication process succeeds. By default, no program is run. The conventional name for a file used here is **Xstartup**. See the Xstartup file section.

systemPath

ClassClass:
SystemPath

Type: String

Default:
system_dep._path

Description:
The **dtlogin** client sets the **PATH** environment variable for the startup and reset scripts to the value of this resource. Note the conspicuous absence of ″.″ from this entry. This is a good practice to follow for root because it avoids many system penetration schemes.

systemShell

ClassClass:
SystemShell

Type: String

Default:
/bin/sh

Description:
The **dtlogin** client sets the **SHELL** environment variable for the startup and reset scripts to the value of this resource.

terminateServer

ClassClass:
TerminateServer

Type: Boolean

Default:
False

Description:
Specifies whether the X server should be terminated when a session ends (instead of resetting it). This option can be used if the server tends to grow indefinitely over time in order to limit the amount of time the server is run continuously.

termSignal

ClassClass:
Signal

Type: Int

Default:
15 (SIGTERM)

Description:
Specifies the signal **dtlogin** sends to terminate the server.

userAuthDir

ClassClass:
UserAuthDir

Type: String

Default:
/var/dt

Description:
When **dtlogin** cannot write to the usual user authorization file (**$HOME/.Xauthority**), it creates a unique file name in this directory and points the environment variable **XAUTHORITY** at the created file.

userPath

ClassClass:
UserPath

Type: String

Default:
system_dep._path

Description:
The **dtlogin** client sets the **PATH** environment variable for the session to this value. It should be a colon-separated list of directories.

xdmMode

ClassClass:
XdmMode

Type: Boolean

Default:
False

Description:
If True, the **$HOME/.xsession** file will be executed from **Xsession** upon user authentication, rather than from **dtsession**.

xrdb

ClassClass:
Xrdb

Type: String

Default:
/system_dep./xrdb

Description:
Specifies the program used to load the resources. The authentication screen reads a *name-password* pair from the keyboard. Because this is a Motif toolkit client, colors, fonts and some layout options can be controlled with resources. General resources for this screen should be put into the file named by the **resources** resource (**Xresources** is the default). Specify language-specific values, such as text or fonts, in the **Dtlogin app-defaults** file.

Logo Resources

The default logo on the authentication screen can be replaced with a bitmap or pixmap of the user's choice. The resources should be prefaced with the string `Dtlogin*logo*` when specified.

bitmapFile

ClassClass:
BitmapFile

Type: String

Default:
NULL

Description:
Specifies the absolute path name to the bitmap or pixmap file to be used for the logo.

background

ClassClass:
Background

Type: Pixel

Default:
#a8a8a8

Description:
Specifies the background color for the logo.

topShadowPixmap

ClassClass:
topShadowPixmap

Type: String

Default:
25_foreground

Description:
Specifies the pixmap to use for the logo border shadow.

The following resources describe the greeting string used on the login screen. The resources should be prefaced with the string `Dtlogin*greeting*` when specified.

foreground

ClassClass:
Foreground

Type: Pixel

Default:
black

Description:
Specifies the foreground color for the welcome message.

background

ClassClass:
Background

Type: Pixel

Default:
dynamic

Description:
Specifies the background color for the welcome message. The default is light gray for color systems or white for monochrome systems.

fontlist

ClassClass:
FontList

Type: FontList

Default:
-*-*schoolbook-medium-i-normal--18-*

Description:
Specifies the font to use for the welcome message.

labelString

ClassClass:
LabelString

Type: String

Default:
Welcome to %LocalHost%

Description:
Specifies the string to use for the welcome message. Multiple lines can be specified by including newline characters (`0` in the text. If the token `%LocalHost%` is included in the text, it will be replaced with the name of the host providing login service. If the token `%DisplayName%` is included in the text, it will be replaced with the display name.

perLabelString

ClassClass:
LabelString

Type: String

Default:
Welcome %s

Description:
Specifies the string to use for the personalized welcome message. This is the message displayed after the user name has been entered. The `%s` will be replaced with the user name entered.

alignment

ClassClass:
Alignment

Type: String

Default:
ALIGNMENT_CENTER

Description:
Specifies the string to use for the alignment of the Welcome message. Valid values are ALIGNMENT_BEGINNING, ALIGNMENT_CENTER and ALIGNMENT_END.

Matte Resources

The following resources describe the matte layout used on the login screen. The resources should be prefaced with the `Dtlogin*matte.` string when specified.

width

ClassClass:
Width

Type: Int

Default:
806 for high-resolution displays
755 for medium-resolution displays
585 for low-resolution displays

Description:
Specifies the width to use for the **login_matte**.

height

ClassClass:
Height

Type: Int

Default:
412 for high-resolution displays
385 for medium-resolution displays
300 for low-resolution displays

Description:
Specifies the height to use for the **login_matte**.

Label Resources

The following resources describe the fonts layout used on the login screen. The resources should be prefaced with the `string Dtlogin*.` when specified.

labelFont

ClassClass:
LabelFont

Type: String

Default:
-*-swiss 742-medium-r-normal-*-140-*-p-110-* for high-resolution displays
-*-swiss 742-bold-r-normal-*-140-*-p-100-* for low-resolution displays

Description:
Specifies the **labelFont** to use for the push buttons and labels.

textFont

ClassClass:
TextFont

Type: String

Default:
-*-prestige-medium-r-normal-*-128-72-* for high-resolution displays
-*-helvetica-bold-r-normal-*-100-* for low-resolution displays

Description:
Specifies the **textFont** to use for the push buttons and labels.

Flags

All flags, except -**config**, specify values that can also be specified in the configuration file as resources. Typically, customization is done using the configuration file rather than command line options. These flags are most useful for debugging and one-shot tests.

-config *configuration_file*	Specifies a resource file that specifies the remaining configuration parameters. This replaces the **dtlogin** default **Xconfig** file. See the Xconfig file section for more information.
-daemon	Specifies `true` as the value for the **daemonMode** resource. This makes **dtlogin** close all file descriptors, disassociate the controlling terminal, and put itself in the background when it first starts up (just like the host of other daemons).
-debug *debug_level*	Specifies the numeric value for the *debug_level* resource. A nonzero value causes **dtlogin** to print debugging statements to the terminal; it also disables the **daemonMode** resource, forcing **dtlogin** to run synchronously.
-error *error_log_file*	Specifies the value for the *error_log_file* resource. See the Xerrors file section for more information.
-nodaemon	Specifies `false` as the value for the resources.
-resources *resource_file*	Specifies the value for the *resource_file* resource. See the the Xresources file section for more information.
-server *server_entry*	Specifies the value for the *server_entry* resource. See the the Xservers file section for more information.
-udpPort *port_number*	Specifies the value for the **requestPort** resource. This sets the port number that **dtlogin** monitors for XDMCP requests. Because XDMCP uses the well-known registered udp port 177, avoid changing this resource except for debugging.
-session *session_program*	Specifies the value for the *session_program* resource. See the Xconfig file section for more information.

Environment Variables

The **dtlogin** command invokes the user's session with the following default environment:

DISPLAY	Set to the associated display name.
EDITOR	Set to **/usr/dt/bin/dtpad**.
HOME	Set to the home directory of the user.
KBD_LANG	Set to the value of **LANG** for applicable languages.
LANG	Set to the current NLS language (if any).
LC_ALL	Set to the current NLS language (if any).
LC_MESSAGES	Set to the current NLS language (if any).

LOGNAME	Set to the user name.
MAIL	Set to **/usr/mail/$USER** (system dependent).
PATH	Set to the value of the **userPath** resource.
USER	Set to the user name.
SHELL	Set to the user's default shell (from **/etc/passwd**).
TERM	Set to **dtterm**.
TZ	Set to the value of the *timeZone* resource or system default.
XAUTHORITY	Set to authority file.

Adding to the Environment List

Four methods are available to modify or add to the preceding list depending on the desired scope of the resulting environment variable:

- The **exportList** resource is available to allow the export of variables provided to the **dtlogin** process by its parent. Variables specified by this method are available to both the display's X server process and the user's session, and they override any default settings. The resource accepts a string of *name*=*value* separated by at least one space or tab.
- The **environment** resource is available in the **dtlogin** configuration file to allow setting of environment variables on a global or per-display basis. Variables specified by this method are available to both the display's X server process and the user's session, and they override any default settings. The resource accepts a string of *name*=*value* separated by at least one space or tab. The values specified must be constants because no shell is used to parse the string. For example:

  ```
  Dtlogin*environment:MAIL_HOST=blanco MAIL_SERVER=pablo
  ```

  Note: The **LANG** and **TZ** environment variables have their own dedicated resources in the configuration file and should not be set by the environment.
- Environment variables that require processing by a shell or are dependent on the value of another environment variable can be specified in the startup script **Xsession**. These variables are loaded into the environment of all users on the display, but not to the X server process. They override any previous settings of the same variable. The **Xsession** script accepts **ksh** syntax for setting environment variables. For example:

  ```
  MAIL=/usr/mail/$USER
  ```
- Personal environment variables can be set on a per-user basis in the **$HOME/.dtprofile** script file. The **dtlogin** command accepts either **sh**, **ksh**, or **csh** syntax for the commands in this file. The commands should only be those that set environment variables, not any that perform terminal I/O, with the exception of **tset** or **stty**. If the first line of **.dtprofile** is `#!/bin/sh`, `#!/bin/ksh` or `#!/bin/csh`, **dtlogin** uses the appropriate shell to parse **.dtprofile**. Otherwise, the user's default shell (**$SHELL**) is used.

Exit Status

The following exit values are returned:

0	Successful completion.
>0	An error occurred.

Examples

1. To start the CDE login service as a daemon, enter:

   ```
   /usr/dt/bin/dtlogin -daemon
   ```
2. To start the CDE login service in debug mode, enter:

   ```
   /usr/dt/bin/dtlogin -debug 1
   ```

Location

/usr/dt/bin/dtlogin

Standard Errors

The **dtlogin** command returns the following error messages:

- `Login incorrect; please try again.`
- `Unable to change to home directory.`
- `Sorry. Maximum number of users already logged in.`
- `Login error, invalid user ID.`
- `Login error, invalid group ID.`
- `Login error, invalid audit ID.`
- `Login error, invalid audit flag.`
- `Logins are currently disabled.`
- `Your current password has expired.`

Files

The **dtlogin** command is designed to operate in a wide variety of environments and provides a suite of configuration files that can be changed to suit a particular system. The default **dtlogin** configuration files can be found in **/usr/dt/config** with the exception of **Xsession**, which is stored in **/usr/dt/bin**. They are as follows:

/usr/dt/config/Xconfig	Specifies other **dtlogin** configuration files and **dtlogin** behavior.
/usr/dt/config/Xaccess	Controls access from displays requesting XDMCP service.
/usr/dt/config/Xservers	Contains the list of displays for **dtlogin** to explicitly manage.
/usr/dt/config/Xresources	Contains resource definitions specifying the appearance of the login screen.
/usr/dt/config/Xsetup	A script executed as root prior to display of the login screen.
/usr/dt/config/Xstartup	A script executed as root after the user has successfully authenticated.
/usr/dt/bin/Xsession	A script executed as the authenticated user that starts the user's session.
/usr/dt/config/Xfailsafe	A script executed as the authenticated user that starts a fail-safe session.
/usr/dt/config/Xreset	A script executed as root after the user's session has exited.

The Xconfig File

The **Xconfig** file contains the general resources for **dtlogin** and is at the top of the **dtlogin** configuration file tree. **Xconfig** specifies the location of other **dtlogin** configuration and log files and specifies **dtlogin** behavior. The location of other **dtlogin** configuration and log files are specified by resource definitions. The defaults are as follows:

Dtlogin.errorLogFile
: /var/dt/Xerrors

Dtlogin.pidFile
: /var/dt/Xpid

Dtlogin.accessFile
: Xaccess

Dtlogin.servers
Xservers

Dtlogin*resources
%L/Xresources

Dtlogin*setup
Xsetup

Dtlogin*startup
Xstartup

Dtlogin*reset
Xreset

Dtlogin*failsafeClient
Xfailsafe

Dtlogin*session
/usr/dt/bin/Xsession

If the path specified for **accessFile**, **servers**, **resources**, **setup**, **startup**, **reset**, **failsafeClient**, or **session** is relative, **dtlogin** will first look for the file in directory **/etc/dt/config**, then **/usr/dt/config**.

Note: Some of the resources are specified with * separating the components. These resources can be made unique for each different display, by replacing the * with the display-name. Refer to Display Resources for more information.

The default **Xconfig** file is **/usr/dt/config/Xconfig**. A system administrator can customize **Xconfig** by copying /**usr/dt/config/Xconfig** to **/etc/dt/config/Xconfig** and modifying **/etc/dt/config/Xconfig**. The default **Xconfig** file contains the preceding configuration and log file entries in addition to a few vendor specific resource definitions and examples.

The Xaccess File

The database file specified by the **accessFile** resource provides information which **dtlogin** uses to control access from displays requesting XDMCP service. This file contains three types of entries: entries which control the response to Direct and Broadcast queries, entries which control the response to Indirect queries, and macro definitions.

The format of a Direct entry is either a host name or a pattern. A pattern is distinguished from a host name by the inclusion of one or more meta characters (* matches any sequence of 0 or more characters, and ? matches any single character) which are compared against the host name of the display device. If the entry is a host name, all comparisons are done using network addresses, so any name which converts to the correct network address can be used. For patterns, only canonical host names are used in the comparison, so ensure that you do not attempt to match aliases. Putting an exclamation point (!) character before either a host name or a pattern causes hosts that match that entry to be excluded.

An Indirect entry also contains a host name or pattern, but follows it with a list of host names or macros to which indirect queries should be sent. Indirect entries can also specify to have **dtlogin** run **dtchooser** to offer a menu of hosts to which a login screen can be displayed.

A macro definition contains a macro name and a list of host names and other macros that the macro expands to. To distinguish macros from host names, macro names start with a % character. Macros can be nested.

When the access for a particular display host is checked, each entry is scanned in turn and the first matching entry determines the response. Direct and Broadcast entries are ignored when scanning for an

Indirect entry and vice-versa. Blank lines are ignored, # is treated as a comment delimiter causing the rest of that line to be ignored, and \newline causes the newline to be ignored, allowing indirect host lists to span multiple lines.

The following example shows an Xaccess file:

```
#
# Xaccess - XDMCP access control file
#

#
# Direct/Broadcast query entries
#
!xtra.lcs.mit.edu # disallow direct/broadcast service for xtra
bambi.ogi.edu     # allow access from this particular display
*.lcs.mit.edu     # allow access from any display in LCS

#
# Indirect query entries
#

#define %HOSTS macro
%HOSTS                expo.lcs.mit.edu xenon.lcs.mit.edu \
                      excess.lcs.mit.edu kanga.lcs.mit.edu

#force extract to contact xenon
extract.lcs.mit.edu xenon.lcs.mit.edu

#disallow indirect access by xtra
!xtra.lcs.mit.edu   dummy

#all others get to choose among %HOSTS
*.lcs.mit.edu       %HOSTS
```

If XDMCP access is granted, a temporary file can be created in the directory specified by **authDir** which contains authorization information for the X-terminal. It is deleted when the session starts.

For X terminals that do not offer a host menu for use with Broadcast or Indirect queries, the **chooser** program can do this for them. In the **Xaccess** file, specify CHOOSER as the first entry in the Indirect host list. The **chooser** program sends a Query request to each of the remaining host names in the list and displays a menu of all the hosts that respond. The list might consist of the word BROADCAST, in which case **chooser** sends a Broadcast instead, again displaying a menu of all hosts that respond. On some operating systems, UDP packets cannot be broadcast, so this feature will not work.

An example of an **Xaccess** file using the **chooser** program is as follows:

```
#offer a menu of these hosts to extract
extract.lcs.mit.edu CHOOSER %HOSTS

#offer a menu of all hosts to xtra
xtra.lcs.mit.edu    CHOOSER BROADCAST
```

The program to use for **chooser** is specified by the **chooser** resource. Resources for this program can be put into the file named by resources. The default **Xaccess** file is **/usr/dt/config/Xaccess**. A system administrator can customize **Xaccess** by copying **/usr/dt/config/Xaccess** to **/etc/dt/config/Xaccess** and then modifying **/etc/dt/config/Xaccess**. The default **Xaccess** file contains no entries.

The Xservers File

The **Xservers** file contains the list of displays to manage. The default **Xservers** file is **/usr/dt/config/Xservers**. A system administrator can customize **Xservers** by copying **/usr/dt/config/Xservers** to **/etc/dt/config/Xservers** and then modifying **/etc/dt/config/Xservers**. The default **Xservers** file contains an entry for one local display.

The Xresources File

The **Xservers** file contains the resource definitions specifying the appearance of the login screen. The default **Xresources** file is **/usr/dt/config/Xresources**. A system administrator can customize **Xresources** by copying **/usr/dt/config/Xresources** to **/etc/dt/config/Xresources** and then modifying **/etc/dt/config/Xresources**.

The Xsetup File

The **Xsetup** file typically a shell script. Only root users can run it, and they should be very careful about security. This script is run before the login screen is displayed. No arguments of any kind are passed to the script. The **dtlogin** command waits until this script exits before displaying the login screen.

The default **Xsetup** file is **/usr/dt/config/Xsetup**. A system administrator can customize **Xsetup** by copying **/usr/dt/config/Xsetup** to **/etc/dt/config/Xsetup** and then modifying **/etc/dt/config/Xsetup**. The default **Xsetup** file contains vendor specific code but typically contains code that sets up the X server prior to the display of the login screen, such as setting up keyboard maps.

The Xstartup File

The **Xstartup** file typically a shell script. Only root users can run it, and they should be very careful about security. This is the place to put commands that display the message of the day or do other system-level functions on behalf of the user. The following environment variables are set for the use of this script:

DISPLAY
: Set to the associated display name.

HOME Set to the home directory of the user.

PATH Set to the value of the **systemPath** resource.

USER Set to the user name.

SHELL
: Set to the value of the **systemShell** resource.

No arguments of any kind are passed to the script. The **dtlogin** command waits until this script exits before starting the user session. If the exit value of this script is nonzero, **dtlogin** discontinues the session immediately and starts another authentication cycle.

The default **Xstartup** file is **/usr/dt/config/Xstartup**. A system administrator can customize **Xstartup** by copying **/usr/dt/config/Xstartup** to **/etc/dt/config/Xstartup** and then modifying **/etc/dt/config/Xstartup**. The default **Xstartup** file contains code to change ownership of **/dev/console** to the user whose session is running on the console.

The Xsession File

The **Xsession** script initializes a user's session and invokes the desktop session manager. It is run with the permissions of the authorized user, and has several environment variables preset. See Environment Variables for a list of the preset variables.

The default **Xsession** file is **/usr/dt/bin/Xsession**. A system administrator can customize **Xsession** by copying **/usr/dt/bin/Xsession** to **/etc/dt/config/Xsession** and then modifying **/etc/dt/config/Xsession**. The session resource defined in **Xconfig** must also be changed to reference the customized **Xsession** file. See The Xconfig File for information on how to update the **Xconfig** file. The default **Xsession** file contains session initialization code. It does contain some vendor specific code, but its general function is as follows:

- Sources the user's **$HOME/.dtprofile**
- Sources any **/etc/dt/config/Xsession.d/*** scripts
- Sources any **/usr/dt/config/Xsession.d/*** scripts
- Launches the desktop welcome client, **dthello**, in the background

- Sources the application search path setup script, **dtsearchpath**
- Launches the help setup client, **dthelpgen**, in the background
- Launches the application manager directory setup client, **dtappgather**, in the background
- Execs the desktop session manager, **dtsession**
-

System administrators are discouraged from customizing the **Xsession** file.

The Xreset File

Symmetrical with **Xstartup**, the **Xreset** script is run after the user session has terminated. Because it is run by a root user, the **Xreset** script should contain commands that undo the effects of commands in **Xstartup**, such as unmounting directories from file servers. The collection of environment variables that were passed to **Xstartup** are also given to **Xreset**.

The default **Xreset** file is **/usr/dt/config/Xreset**. A system administrator can customize **Xreset** by copying **/usr/dt/config/Xreset** to **/etc/dt/config/Xreset** and then modifying **/etc/dt/config/Xreset**. The default **Xreset** file contains code change ownership of **/dev/console** back to root.

The Xerrors File

The **Xerrors** script contains error messages from **dtlogin** and anything output to **stderr** by **Xsetup**, **Xstartup** or **Xreset**. The system administrator can use the contents of this file for **dtlogin** troubleshooting. The **errorLogSize** resource limits the size of the **Xerrors** file and can prevent it from growing without bound. If the file does grow larger than the requested size and is truncated by **dtlogin**, any user who is accessing the file (for example, using **cat** or **tail**) will need to close the file (after the file is truncated) and reopen it for access in order to see subsequent information that is logged to the file.

A system administrator can change the path name of the **Xerrors** by setting the **errorLogFile** resource in the **Xconfig** file.

The Xpid File

The **Xpid** script contains the process ID of the master **dtlogin** process, which can be used when sending signals to **dtlogin**. A system administrator can change the path name of the **Xpid** by setting the **pidFile** resource in the **Xconfig** file.

Related Information

The “dtaction Command” on page 210, “dtlogin Command” on page 213.

---

dtscript Command

Purpose

Builds simple dialogs used in the X Window System environment.

Syntax

dtscript [**-xrm** *options*] [**-dir** *Path*] [**-file** *FileName*] [**-workspace** *WorkspaceName*]

> **Note:** The **-xrm** *options* must be specified, if used, before any other flag.

Description

Desktop Script supports a subset of Motif widgets you drag and drop from the palette into your dialog. You can move or resize any widget in a dialog. You can also edit widget properties using the specialized editors provided.

You can enter callbacks to give widgets desired behavior. When a dialog is complete, Desktop Script generates dtksh code for it.

Flags

-dir *Path*	Sets Desktop Script's current directory shown in the File Select dialog to *Path*.
-file *FileName*	Loads an existing dialog called: *FileName*. The *FileName* argument can be an absolute path name, a path name relative to the current directory, or a path name relative to the **-dir** value.
-workspace *WorkspaceName*	Loads Desktop Script into the corresponding CDE workspace.
-xrm *options*	Enables you to enter any of the specifications (*options*) that you would otherwise put into a resource file.

Examples

To invoke the Desktop Script from a window, enter:

```
dtscript
```

Files

/usr/dt/bin/dtscript	Contains the **dtscript** command.

dtsession Command

Purpose

Manages a CDE session.

Syntax

dtsession [*options*] ...

Description

The **dtsession** command provides session management functionality, compliant with ICCCM 1.1, during a user's session, from login to logout. It launches a window manager and allows users to save a session, restore a session, lock a session, launch screen savers, and allocate colors for desktop-compatible clients.

Note: The desktop login manager **dtlogin** automatically invokes the **dtsession** client through the **Xsession** script. The **dtsession** client can also be started through the **Xsession** script on an existing X server. The **dtsession** session manager automatically starts a window manager.

The **dtsession** command supports the following tasks:

- Initializing a session
- Launching a window manager
- Restoring a home or current session
- Providing session locking on command or timeout
- Providing session screen saving on command or timeout
- Acting as a color allocation server for other desktop clients
- Saving a home or current session
- Displaying confirmation dialog at logout
- Displaying session selection dialog at logout

- Terminating a session

Sessions

A session is the collection of applications, settings and resources that are present on the user's desktop. Session management is a set of conventions and protocols that allow a special session manager, such as **dtsession**, to save and restore a user's session. A user can log in to a system and be presented with the same set of running applications, settings, and resources that were present when the user logged off. When a user logs in to the desktop for the first time, a default initial session is loaded. Afterward, **dtsession** supports the notion of a current and a home session.

The following sessions are defined:

Initial Session
When a user logs in to the desktop for the first time, **dtsession** generates the user's initial session using system default values. Refer to Session Resource Management and Session Application Management for more information.

Current Session
The user's session that is running is always considered the current session, whether restored upon login from a saved home session, a saved current session, or the system default initial session. Based on the user's Style Manager Startup settings, when the user exits the session, the current session is automatically saved. When the user next logs in to the desktop, the previously saved current session is restarted. The desktop is restored to the same state it was in when the user last logged out.

Home Session
Another option restores the desktop to the same state every time the user logs in, regardless of its state when the user logged out. The user can save the state of the current session, then sets the Style Manager Startup so that the desktop starts that session every time the user logs in.

Display-Specific Sessions
To run a specific session for a specific display, users can create a display-specific session. To do this, users can copy the **$HOME/.dt/sessions** directory to **$HOME/.dt/***display*, where display is the real, unqualified host name (for example, `pablo:0` is valid, but pablo.gato.com:0 or `local:0` is not). When the user logs in on display `pablo:0`, that display-specific session takes precedence.

ICCCM Session Management Protocol

For an application to be saved upon logout and restarted upon login, it must participate in a simple session management protocol. The **dtsession** command supports the ICCCM 1.1 Session Management Protocol.

Applications that want to save their state can take part in the WM_SAVE_YOURSELF protocol. To do this, an application needs to set the WM_SAVE_YOURSELF property on one and only one of its top-level windows. When a session is saved, **dtsession** sends the application's top-level window a WM_SAVE_YOURSELF client message. At this point, the application proceeds to quietly save its state. The application cannot interact with the user in any way while it is saving its state. Because an application will likely save its state into a file, the session manager provides a convenience function, **DtSessionSavePath**, which returns a full path name of a file in which an application can save its state. While the application is saving its state, **dtsession** awaits notice from the application that it is finished. In order to tell **dtsession** that the state save is complete, the application must update the WM_COMMAND property on its top-level window.

The WM_COMMAND property on an application's top-level window serves two purposes. First, a change of this property indicates to **dtsession** that an application is finished saving its state and **dtsession** can proceed to the next application. Second, the WM_COMMAND property value is expected to contain the command line that **dtsession** uses to restart the application at session startup. If an application is

launched with a full path name, it should use the full path name when setting WM_COMMAND. Applications that do not need to save their state but want to be restarted can simply set WM_COMMAND once during application startup.

Restoring a Session

At session startup time, **dtsession** determines which session to restore. The following list describes the order of precedence:

1. Display-specific Current of Home Session
2. Current or Home Session
3. Initial Session

Session Resource Management

The session manager uses the X Server RESOURCE_MANAGER property on which to make available desktop resources to all applications. The session manager loads the RESOURCE_MANAGER in the following manner:

1. Loads the system default resources.
2. Merges any system administrator-specified resources.
3. Merges any user-specified resources.

The desktop default resources can be found in the **/usr/dt/config/$LANG/sys.resources** file. These resources are made available to each user's session through the RESOURCE_MANAGER property. Do not edit this file because it is unconditionally overwritten during subsequent desktop installations.

By creating a **/etc/dt/config/$LANG/sys.resources** file, a system administrator can override system default resources or specify additional resources. Because this file is merged into the desktop default resources during session startup, only new or updated resource specifications should be placed in this file. This is preferable to making a copy of the desktop default resource file. Resources specified in this file are made available to each user’s session through the RESOURCE_MANAGER property. Resources specified in this file take precedence over those specified in the desktop default resource file.

By editing the **$HOME/.Xdefaults** file, a user can override the desktop default and system administrator resources. Resources specified in this file are made available to only that user's session through the RESOURCE_MANAGER property and take precedence over those resources specified in the desktop default or system administrator resource files.

Note: The X Toolkit Intrinsics specifies that it will load application resources from either RESOURCE_MANAGER or from **$HOME/.Xdefaults**, but not both. Ordinarily, this means that the user's **$HOME/.Xdefaults** file would be ignored. However, the session manager accommodates **$HOME/.Xdefaults** by merging it into the RESOURCE_MANAGER at session startup, as previously described. When users change their **$HOME/.Xdefaults** files, their changes are not visible to new applications until the users invoke the **ReloadResources** action.

The **ReloadResources** action instructs the session manager to reload the RESOURCE_MANAGER with the system-specified, system administrator-specified,, and user-specified resources. This makes available to new applications changes that were made to system administrator-specified or user-specified resource files.

Session Application Management

At session startup, the session manager restarts any applications that were saved as part of the session. The system default set of applications to be restored as part of the user's Initial Session can be found in the **/usr/dt/config/$LANG/sys.session** file. Do not edit this file because it is unconditionally overwritten during subsequent desktop installations.

A system administrator can replace the set of applications that are restored as part of the user's Initial Session by creating a **/etc/dt/config/$LANG/sys.session** file. Unlike the resource files, this file is used as a complete replacement for the desktop default file, so you can make a copy of the system default file and make any necessary modifications.

The Window Manager

The **dtsession** command starts the window manager. By default, **/usr/dt/bin/dtwm** is started. An alternate window manager can be specified using the **wmStartupCommand** resource. Refer to the Workspace Manager specification for more information.

The Style Manager

The style manager provides the interface by which a user can change various desktop and X server settings for the current session. Refer to the Style Manager specification for more information.

The Color Server

The **dtsession** command serves as the color server for the desktop and provides the following set of resources that can be used to configure it:

foregroundColor
Controls whether a pixel is allocated for the foreground color.

dynamicColor
Specifies whether read-only colors are allocated.

shadowPixmaps
Specifies whether colors are allocated for top shadow or bottom shadow.

colorUse
Limits color allocation.

writeXrdbColors
Specifies whether the *background and *foreground resources are placed in the resource database.

See the Color Server Resources section for more information.

Session Lock

The **dtsession** command provides session locking. The current session can be locked directly by pressing the lock icon on the front panel. If supported by the X server, the current session can be locked after a specified period of inactivity. To unlock the session, users must enter their login password, the login password for the root user, or the login password for any of the users specified by the **keys** resource. See Screen Lock and Screen Save Resources for more information on the keys resource.

The **dtsession** command is a PAM-enabled session manager with service name **dtsession**. It supports traditional local UNIX authentication as well as PAM authentication for unlocking the session. Additional reauthentication functionality, such as that required by DCE, can be added by individual vendors.

System-wide configuration to use PAM for authentication is set by establishing root user permissions and modifying the value of the *auth_type* attribute in the **usw** stanza of the **/etc/security/login.cfg** file to `PAM_AUTH`.

The authentication mechanisms used when PAM is enabled depend on the configuration for the login service in **/etc/pam.conf**. The **dtsession** command requires an **/etc/pam.conf** entry for the **auth** module type. The following configuration is recommended in **/etc/pam.conf** for the **dtsession** service:

```
dtsession       auth            required       /usr/lib/security/pam_aix
```

Screen Savers

The **dtsession** command provides support for the launching of external screen savers as a part of session locking from the front panel or, if supported by the X server, after a specified period of inactivity. Refer to the Screen Saver specification for information as to how screen savers are integrated into the desktop.

X Server Screen Saver Extensions

The **dtsession** command's ability to provide session lock or screen saver launch after a specified period of inactivity depends upon the availability of an X server screen saver extension. The **dtsession** command supports the X Consortium Sample X11 Screen Saver Extension 1.0 and the HP X Screen Saver Extension. The ability of the **dtsession** command to recognize both, either, or none of these extensions is vendor specific.

Launching the Session Manager

The **dtsession** command should be launched from the **Xsession** script. **Xsession** is described in the login manager specification. Although launching **Xsession** from **dtlogin** as part of the default login sequence is recommended, some systems allow proxy programs, such as xinit, x11start, or startx, to start **Xsession**.

Color Server Resources

colorUse

ClassClass:
ColorUse

Type: String

Default:
DEFAULT

Description:
Specifies the number of colors to use for the user interface. Color server will determine type of monitor based upon number of display planes of the screen as follows:

1,2 or 3 planes (B_W)
Specifies a black-and-white system. The color palettes use two color cells for the user interface. In this configuration, only two color palettes are available: BlackWhite and WhiteBlack. These palettes cannot dynamically change. To change a palette, all applications using that color palette must be restarted. This resource value forces **ShadowPixmaps** to True, and **ForegroundColor** to either black or white (depending on the palette chosen).

4 or 5 planes (LOW_COLOR)
Specifies a low-color system. The color palettes have two color sets and use a maximum of 12 color cells for the user interface, including black and white (color cells 0 and 1). The number of color cells can be reduced by using the resources **ShadowPixmaps** and **ForegroundColor**.

6 planes (MEDIUM_COLOR)
Specifies a medium-color system. The color palettes have four color sets and use a maximum of 22 color cells for the user interface, including black and white (color cells 0 and 1). The number of color cells can be reduced by using the resources **ShadowPixmaps** and **ForegroundColor**.

7+ planes (HIGH_COLOR)
Specifies a high-color system. The color palettes have eight color sets and use a maximum of 42 color cells for the user interface, including black and white (color cells 0 and 1). The number of color cells can be reduced by using the resources **ShadowPixmaps** and **ForegroundColor**.

dynamicColor

ClassClass:
DynamicColor

Type: Boolean

Default:
True

Description:
This resource can have values of True or False. The **dynamicColor** resource is used to reduce the number of color cells being used. After a palette has been selected and it is not likely to be changed, **dynamicColor** can be set to False. If set to False, colors cannot be dynamically changed using the desktop style manager. A selected palette takes effect the next session. The next time the session comes up, the color server uses Read Only color cells that can be shared by all clients, reducing the number of color cells used.

foregroundColor

ClassClass:
ForegroundColor

Type: String

Default:
DYNAMIC

Description:
This resource can have values of White, Black, or Dynamic. The **foregroundColor** resource causes all text (foreground) to use either pixel 0 or 1 (Black or White) or to have a color cell dedicated to foreground and changes in response to the background color (Dynamic) for each ColorSet. If set to White or Black, the number of color cells used per ColorSet is reduced by 1.

shadowPixmaps

ClassClass:
ShadowPixmaps

Type: String

Default:
DEFAULT

Description:
For color systems, this resource can have a value of True or False. If True, **topShadowColor** and **bottomShadowColor** use the same pixel as background and **topShadowPixmap** and **bottomShadowPixmap** are specified instead of solid color to create the 3-D look. This reduces the number of color cells per ColorSet by 2. This resource defaults to True for systems with four or less color planes (16 or less color cells), and False for systems with more than four color planes.

writeXrdbColors

ClassClass:
WriteXrdbColors

Type: Boolean

Default:
True

Screen Lock and Screen Save Resources

keys

ClassClass:
Keys

Type: unsigned char

Default:
NULL

Description:
Lists key holders who have the ability to unlock the screen any time it is locked by the user. The list is a list of user IDs separated by commas. For example, if user `kim` has the following resource active during a session, users `fred` and `keith` have the ability to unlock the display when `kim` locks it:

```
Dtsession*keys: fred,keith
```

passwordTimeout

ClassClass:
passwordTimeout

Type: unsigned int

Default:
10

Description:
Specifies (in seconds) the amount of time before the password dialog is removed from the screen. When the display is locked, the pointer shows a lock cursor, and a dialog is displayed that asks for the user password. If no activity from the pointer or keyboard is detected for *passwordTimeout* seconds, the dialog is removed from the screen. The dialog is redisplayed as soon as a pointer or keyboard event is detected. A *passwordTimeout* of 0 leaves the password dialog in place for the entire time the display is locked. The default value is 10 seconds.

Miscellaneous Resources

queryServerSettings

ClassClass:
QueryServerSettings

Type: Boolean

Default:
False

Description:
Specifies whether the **dtsession** command queries the server at logout for all its settings, or whether it saves only those settings set by using the desktop Style Manager. Querying the server ensures that all settings are saved; however, there is a degradation in performance when a full query is done. The default value is False, which means that the server will not be queried.

saveFontPath

ClassClass:
SaveFontPath

Type: Boolean

Default:
False

wmStartupCommand

ClassClass:
WmStartupCommand

Type: executable path

Default:
NULL

Description:
Allows for an alternate window manager to be started at login. If this resource is NULL, **dtsession** starts **/usr/dt/bin/dtwm**. An alternate startup might look like:

```
Dtsession*wmStartupCommand:   /usr/bin/X11/mwm
```

The command should not have any commands to a shell in it, and it should not be surrounded by quotes. If any other window manager other than **/usr/dt/bin/dtwm** is used, clients will be restored but might not be restored to the correct position. By default, this resource contains a NULL value.

Flags

-migrate Instructs **dtsession** to migrate the resource information saved from a previous session. If this option is specified, client restart information from a previously saved session might be ignored. This option should only be required if the previous session was saved on AIX 4.1.1 or AIX 4.1.2, and would normally be specified by modifying the **/usr/dt/bin/Xsession** script. The **/usr/dt/bin/Xsession** script contains information about how to modify that script to specify this option for **dtsession**.

Exit Status

The following exit values are returned:

0	Successful completion.
>0	An error occurred.

Examples

1. To start the session manager from the command line without restoring the previous session, enter:
   ```
   dtsession -norestore
   ```

Location

/usr/dt/bin/dtsession

Files

/usr/dt/config/$LANG/sys.session	The desktop default set of applications for the user's Initial Session.
/etc/dt/config/$LANG/sys.session	System administrator-specified set of applications for the user's Initial Session.
/usr/dt/config/$LANG/sys.resources	The desktop default resources.
/etc/dt/config/$LANG/sys.resources	The system administrator-specified resources.
$HOME/.Xdefaults	The user-specified resources. **Note:** The **dtsession** command stores session information in **$HOME/.dt/display** or **$HOME/.dt/sessions**. The content of these directories should not be directly edited by the user.
/usr/dt/app-defaults/$LANG/Dtsession	Default **dtsession** resources.

Related Information

The “dtaction Command” on page 210, “dtlogin Command” on page 213.

dtterm Command

Purpose

Provides runtime support of legacy applications.

Syntax

dtterm [Flags...]

Description

The **dtterm** client provides runtime support of legacy applications written for ANSI X3.64-1979 and ISO 6429:1992(E) conformant character terminals.

Flags

Note: The **dtterm** terminal emulator accepts all of the standard X Toolkit command line flags along with additional flags, all of which are listed below (if the flag begins with a **+** instead of a **-**, the flag is restored to its default value):

-132 — Causes the DECCOLM escape sequence to be recognized, and the **dtterm** window will resize appropriately. Normally the DECCOLM escape sequence that switches between 80 and 132 column mode is ignored. Associated resource: c132.

+132 — Causes the DECCOLM escape sequence to be ignored. This is the default behavior. Associated resource: c132.

-aw — Indicates that auto-wraparound should be allowed. This allows the cursor to automatically wrap to the beginning of the next line when it is at the right-most position of a line and text is output. This is the default behavior. Associated resource: autoWrap.

+aw — Indicates that auto-wraparound should not be allowed. Associated resource: autoWrap.

-background *background_color* — Specifies the background of the terminal window as well as the default background used for the scroll bar and the X11 pointer cursor. Under CDE, this flag defaults to the primary colorset select pixel or background pixel, see -bs. Without CDE, this flag defaults to *background/*Background with an ultimate fallback color of black. *background_color* describes the background color to use. Associated resource: background.

-bd *border_color* — Specifies the border color for all windows. The shell widget's border may not be visible when reparenting window managers such as **dtwm** and **mwm** are used. The default color is black. *border_color* describes the border color to use. Associated resource: borderColor.

-bg *background_color* — Identical to -background. *background_color* describes the background color to use. Associated resource: background.

-bordercolor *border_color* — Identical to -bd above. *border_color* describes the border color to use. Associated resource: borderColor.

-borderwidth *border_width* — Specifies the border width of the shell widget's window. This value may be overridden by reparenting window managers such as **dtwm** and **mwm**. The default is 0. *border_width* specifies the width of the window border in pixels. Associated resource: borderWidth.

-bs — Specifies that the terminal window should use the Motif select color instead of the background color for the terminal window's background color. This is the default behavior. Associated resource: backgroundIsSelect.

+bs — Specifies that the terminal window should not use the Motif select color instead of the background color for the terminal window's background color. Associated resource: backgroundIsSelect.

-bw *border_width* — Identical to -borderwidth. Associated resource: borderWidth.

-C — Specifies that output directed at **/dev/console** should be directed instead to the terminal window. It is provided as a way to prevent output that would normally be displayed on the ITE from overwriting the X server's display. It is not provided as a general mechanism to direct the output from an arbitrary system's **/dev/console** to an arbitrary X server.

Note: You must have ownership of and read/write access to **/dev/console** for this flag to work.

-display *display_name* — Specifies the X11 display server to be used by **dtterm**. This defaults to the value in the $DISPLAY environment variable. *display_name* specifies the X11 server to connect to.

-e *program_argument...* Specifies an executable program to be invoked as a subprocess when **dtterm** is started. This flag must be the last flag on the command line. *program_argument* specifies the program and command line arguments to run.

-fb *fontset* Specifies an XFontSet to be used when displaying bold terminal text. It should be specified as a Motif XmFontList. Only character or mono spaced fonts are supported. The behavior when using proportional fonts is undefined. A default bold font will be generated based on the XLFD name of the userFont. If that font is not available, bold text will be generated by overstriking (with a one pixel offset) the userFont. *fontset* specifies the bold terminal XFontSet to use. Associated resource: userFont.

-fg *foreground_color* Specifies the foreground color of the terminal window as well as the default foreground color used for the scroll bar and for the X11 pointer cursor. Under CDE, this resource will default to the primary color set foreground pixel. Without CDE, this resource will default to *foreground or *Foreground with an ultimate fallback color of white. *foreground_color* specifies the foreground color to use. Associated resource: foreground.

-fn *fontset* Specifies an XFontSet to be used when displaying terminal text. It should be specified as a Motif XmFontList. Only character or mono spaced fonts are supported. The behavior when using proportional fonts is undefined. This font will not be used to display non-terminal text (menu bar, popup menus, dialogs, etc.). The default is to use the XmNtextFontList value of the parent bulletin board (see XmBulletinBoard) in the same manner as the XmText widget. *fontset* specifies the terminal XFontSet to use. Associated resource: userFont.

-font *fontset* Identical to -fn. *fontset* specifies the terminal XFontSet to use. Associated resource: userFont.

-foreground *foreground* Identical to -fg. *foreground* specifies the foreground color to use. Associated resource: foreground.

-geometry *geometry_string* Specifies the preferred size and position of the terminal window. The default size is 24 lines of 80 characters each. There is no default position. *geometry_string* specifies the terminal geometry to use. Associated resource: geometry.

-help Displays a message summarizing the usage of **dtterm**.

-iconic Specifies that the terminal emulator should initially be placed on the display iconified. Associated resource: iconic.

+iconic Specifies that the terminal emulator should initially be placed on the display as a normal window. This is the default behavior. Associated resource: iconic.

-j Specifies that jump scrolling should be used. Under jump scrolling, the screen may be scrolled more than one line at a time. This provides for faster screen updates when multiple lines of text are being sent to the terminal. The maximum number of lines that may be jump scrolled is limited to the number of lines in the terminal window. All lines are displayed. This is the default behavior. Associated resource: jumpScroll.

+j Specifies that jump scrolling should not be used. For a description of jump scrolling, see -j. Associated resource: jumpScroll.

-kshMode Specifies that **ksh** mode should be enabled. Under **ksh** mode, a key pressed with the extend modifier bit set will generate an escape character followed by the character generated by the un-extended keystroke. This flag is provided for use with emacs and the emacs command line editor mode of **ksh** or **ied**. It conflicts with **** the normal use of the meta key for generating extended single byte characters, and for generating multi-byte Asian characters. Associated resource: kshMode.

+kshMode Specifies that the **ksh** mode should not be enabled. This is the default behavior. Associated resource: kshMode.

-l Enables output logging. When logging is enabled, all output received from the subprocess is logged either to a file or to a command pipeline (as specified via the -lf flag). Since the data is being logged directly from the subprocess, it includes all escape characters and carriage return/newline pairs sent by the terminal line discipline. Output may be enabled and disabled via escape sequences. Associated resource: logging.

+l Disables output logging. For a description of output logging, see -l. This flag is the default. Associated resource: logging.

-lf *file_name*
Specifies the name of the file to which the output log described in the -l flag. If *file_name* begins with a pipe symbol (|), the rest of the string is assumed to be a command to be used as the endpoint of a pipe. The default filename is DttermLog*XXXXX* (where *XXXXX* is the process id of **dtterm**) and is created in the directory from which **dtterm** was started. If the last five characters are *XXXXX*, they are replaced by the process ID. *file_name* specifies the log file name to use. Associated resource: logFile.

-ls
Indicates that the shell that is started should be a login shell (i.e. the first character of argv[0] will be a dash, indicating to the shell that it should read the system's profile and the user's **$HOME/.profile** (for **ksh** and **sh**) or the system's **csh.login** and the user's **$HOME.login** (for **csh**). Associated resource: loginShell.

+ls
Specifies that a normal (non-login) shell should be started. This is the default behavior. Associated resource: loginShell.

-map
Indicates that **dtterm** should map (de-iconify) itself upon subprocess output if it is unmapped (iconified). An initial period of time during which **dtterm** will not map itself upon subprocess output may be specified via the mapOnOutputDelay resource. Associated resource: mapOnOutput.

+map
Specifies that there should be no special mapping behavior. This is the default behavior. Associated resource: mapOnOutput.

-mb
Indicates that **dtterm** should ring a margin bell when the user types near the right margin. The actual distance involved is specified by the -nb flag. Associated resource: marginBell.

+mb
Indicates that margin bell should not be rung when the user types near the right margin. This is the default. Associated resource: marginBell.

-ms *pointer_color*
Specifies the foreground color to use for the terminal window's (X11) pointer cursor. The default is to use the terminal window's foreground color. See foreground. *pointer_color* specifies the pointer foreground color to use. Associated resource: pointerColor.

-name *prog_name*
Specifies the X11 name of the **dtterm** window. *prog_name* the name to use.

-nb *number*
Specifies the number of characters from the right margin at which the margin bell will ring, if enabled. The default is 10. Associated resource: nMarginBell.

-r
Causes the **dtterm** window to be displayed with the foreground and background colors reversed. This is identical to the -rv and -reverse flags.

+r
Causes the **dtterm** window to be displayed with the normal foreground and background colors. This is the default, and is also identical to the +rv flag.

-reverse
Causes the **dtterm** window to be displayed with the foreground and background colors reversed. This is identical to the -r and -rv flag.

-rv
Causes the **dtterm** window to be displayed with the foreground and background colors reversed. This is identical to choosing Options | Global Options, and then changing the ``windowBackground'' options menu to ``Inverse.'' A **dtterm** window started with this flag has the`` Window Background'' options menu set to ``Inverse.'' See ``Global Options''.

+rv
Causes the **dtterm** window to be displayed with the normal foreground and background colors. This is the default.

-rw
Specifies that reverse-wraparound should be enabled. Associated resource: reverseWrap.

+rw
Indicates that reverse-wraparound should not be enabled. This is the default. Associated resource: reverseWrap.

-Sccn
Specifies that the terminal emulator should be run against a pre-opened pty or STREAMS device. This flag is provided for use where the pty or STREAMS device's slave name is of the form tty?? (i.e., exactly two characters following the tty). This flag is intended for use when **dtterm** is invoked programmatically from another application. *cc* specifies the last two characters of the pty or STREAMS device's slave name, where the slave name is of the form tty??. This value is ignored, but must be exactly two characters in length. *n* specifies the number of the file descriptor that corresponds to the pty or STREAMS device's already-opened master side.

-Sc.n This flag is identical to **-Sccn** above, but is provided for systems with a larger pty name space. *c* specifies the last component of the pty slave name. This values is ignored and may be empty. *n* specifies the number of the file descriptor that corresponds to the pty's already-opened master side.

-sb Indicates that a scrollbar should be displayed. This is the default. Associated resource: scrollBar.

+sb Indicates that a scrollbar should not be displayed. Associated resource: scrollBar.

-sf Indicates that Sun Function Key escape codes should be generated for function keys instead of standard VT220 escape sequences. Associated resource: sunFunctionKeys.

+sf Indicates that the standard escape sequences should be generated for function keys instead of the Sun Function Key escape codes. This is the default behavior. Associated resource: sunFunctionKeys.

-sl *screens[sll]* Specifies the number of lines in the terminal buffer beyond the length of the window. The flag value consists of a number followed by an optional suffix. If no suffix is included, or the suffix is **l** (ell), the total length of the terminal buffer will be screens plus the length of the terminal window. If the suffix is **s** (ess), the total length of the terminal buffer will be (screens plus one) times the length of the terminal window. **dtterm** will try to maintain the same buffer-to-window ratio when the window is resized larger. The default is **4s**. *screens* specifies the number of screens or lines to save. Associated resource: saveLines.

-ti *term_id* Supplies the name used to select the correct response to terminal ID queries. Valid values are vt100, vt101, vt102, and vt220. The default is vt220. *term_id* specifies the terminal ID to use.

-title *title_string* Specifies the window title. If the -e flag is used, the default will be the last component of the program's path. If the -e flag is not used, the default will be the last component of the name used to run **dtterm** (i.e., argv[0]). *title_string* specifies the title to use. Associated resource: title.

-tm *term_modes* Specifies a string containing terminal-setting keywords and the characters to which they may be bound. Allowable keywords include intr, quit, erase, kill, eof, eol, swtch, start, stop, brk, susp, dsusp, rprnt, flush, weras, and lnext. Keywords that do not apply to a specific architecture will be correctly parsed and ignored. Control characters may be specified as ^ followed by char (e.g. ^c or ^u), and ^? may be used to indicate delete. This is useful for overridding the default terminal settings without having to do an **stty** every time a terminal process is started. The default is NULL. *term_modes* specifies the terminal mode string. Associated resource: ttyModes.

-tn *term_name* Specifies a name to set the $TERM environment variable to. The default is **vt220**. *term_name* specifies the terminal name to use. Associated resource: termName.

-usage Prints a usage message on the screen.

-vb Indicates that a visual bell is preferred over an audible one. Instead of ringing the terminal bell whenever a Control-G is received, the window will be flashed. Associated resource: visualBell.

+vb Indicates that an audio bell is preferred over a visual one. This is the default behavior. Associated resource: visualBell.

-w *border_width* Identical to -borderwidth. *border_width* specifies the width of the window border in pixels.

-xrm *resource_string* Allows X11 Resource Manager-style resources to be specified on the command line. *resource_string* specifies an X11 resource string.

Resources

allowSendEvents Specifies that the terminal emulator should allow synthetic events (generated and sent by another application). Enabling this resource opens up a possible security risk. The default is False.

appCursorDefault If True, the cursor keys are initially in application mode. If False, they are initially in cursor mode. The default is False.

appKeypadDefault If True, the keypad keys are initially in application mode. If False, they are initially in numeric mode. The default is False.

autoWrap Specifies whether or not auto-wraparound is initially enabled. The default is True.

background Specifies the background color of the terminal window as well as the default background color used for the scrollbar. Under CDE, this resource defaults to either the primary color set select pixel or the primary color set background pixe, see backgroundIsSelect. The default is the primary color set background pixel. Without CDE, this resource defaults to black.

backgroundIsSelect When True, this resource specifies that the terminal window should use the Motif select color instead of the background color for the terminal window's background color. The default is False.

blinkRate Specifies the number of milliseconds the cursor is in the on and off states while blinking. A value of 250 will blink the cursor two times per second. A value of 0 will turn blinking off. The default is 250.

borderColor Defines the border color for the window. The window border may not be visible when reparenting window managers such as **dtwm** and **mwm** are used. The default is ``black''.

borderWidth Specifies the border width of the shell widget's window. This value may be overridden by reparenting window managers such as **dtwm** and **mwm**. The default is 0.

c132 Specifies whether or not the DECCOLM escape sequence that switches to window with between 80 and 132 columns should be honored. The default is False.

charCursorStyle Specifies the shape of the text cursor. A value of char_cursor_box specifies a cursor with the width and height of the base font's bounding box. A value of char_cursor_bar specifies a cursor with the width of the base font's bounding box, a height of two pixels, and drawn with it's top on the baseline. The default is char_cursor_box.

consoleMode Specifies that output directed at **/dev/console** should be directed instead to the terminal window. It is provided as a way to prevent output that would normally be displayed on the ITE from overwriting the X server's display. It is not provided as a general mechanism to direct the output from an arbitrary system's **/dev/console** to an arbitrary X server. Note that you must have ownership of and read/write access to **/dev/console** for this flag to work. The default is False.

foreground Specifies the foreground color of the terminal window as well as the default foreground color used for the scrollbar and the color used for the pointer cursor. Under CDE, this resource will default to the primary colorset foreground. Otherwise, it defaults to ``white''.

geometry Specifies the preferred size and position of the terminal window. The default size is 24 lines of 80 characters each. There is no default position.

iconGeometry Specifies the preferred position of the terminal emulator's icon. Window managers may ignore this value. There is no default.

iconic If true, specifies that the terminal emulator should initially be placed on the display iconified. Window managers (including **dtwm** and **mwm** may ignore this value. The default is False.

iconicName Specifies the name for the icon. If the -e flag is used, the default will be the last component of the program's path. If the -e flag is not used, the default will be the base name of the name used to run **dtterm** (i.e., argv[0]).

jumpScroll Specifies that jump scrolling should be used. Under jump scrolling, the screen may be scrolled more than one line at a time. This provides for faster screen updates when multiple lines of text are being sent to the terminal. The maximum number of lines that may be jump scrolled is limited to the number of lines in the display. It is guaranteed that all lines will be displayed. The default is True.

kshMode Specifies that **ksh** mode should be enabled. Under **ksh** mode, a key pressed with the extend modifier bit set will generate an escape character followed by the character generated by the un- extended keystroke. This flag is provided for use with emacs and emacs command line editor mode of **ksh** or **ied**. It conflicts with the normal use of the meta key for generating extended single byte characters and for generating multi-byte Asian characters. The default is False.

logFile Specifies the name of the file to which the output log described below is written. If the filename begins with a pipe symbol (|), the rest of the string is assumed to be a command to be used as the endpoint of a pipe. The default filename is DttermLog*XXXXX* (where *XXXXX* is a unique character string) and is created in the directory from which the subprocess was started. If the last five characters are *XXXXX*, they are replaced by a unique character string.

logging Enables output logging. When logging is enabled, all output received from the subprocess is logged either to a file or to a command pipeline (as specified via the logFile flag). Since the data is being logged directly from the subprocess, it includes all escape characters and carriage return/newline pairs sent by the terminal line discipline. Output may be enabled and disabled via escape sequences. The default is False.

logInhibit Specifies that device and file logging should be inhibited. The default is False.

loginShell Specifies that the shell that is started should be a login shell (i.e. the first character of argv[0] will be a dash, indicating to the shell that it should read the system's profile and the user's **$HOME/.profile** (for **ksh** and **sh**) or the system's **csh.login** and the user's **$HOME/.login** (for **csh**). The default is False.

mapOnOutput Indicates that the terminal emulator should map (de-iconify) itself upon subprocess output if it is unmapped (iconified). An initial period of time during which it will not map itself upon subprocess output may be specified via the mapOnOutputDelay resource. The default is False.

mapOnOutputDelay Specifies the number of seconds after start-up that **dtterm** will not honor the mapOnOutput resource. This allows for initial output (e.g., shell prompts) to be sent to the terminal without auto mapping the window. The default is 0 (no delay)

marginBell Specifies whether or not the bell should be run when the user types near the right margin. The default is False.

menuBar Specifies that a pulldown menu should be displayed. The default is True.

menuPopup Specifies that a popup menu should be enabled. The default is True.

nMarginBell Specifies the number of characters from the right margin at which the margin bell should be rung, when enabled. The default is 10.

pointerBlank Specifies that the pointer cursor should be put into blanking mode. In this mode, the cursor will turn on when the pointer is moved, and will be blanked either after a selectable number of seconds or after keyboard input has occurred. The delay is set via the pointerBlankDelay resource. The default is False.

pointerBlankDelay Defines the number of seconds to wait before blanking the pointer cursor after the pointer has been moved. A value of 0 invokes pointer blanking only on keyboard input. The default is 2 seconds.

pointerColor Specifies the foreground color to use for the terminal window's pointer (X11) cursor. The default is to use the terminal window's foreground color. See foreground.

pointerColorBackground Specifies the background color to use for the terminal windows pointer (X11) cursor. The default is to use the terminal windows background color See background.

pointerShape Specifies the X cursor font character to use as the pointer cursor. It should be specified as a string from the include file with the leading XC_ removed. The default is **xterm**.

reverseVideo Specifies whether or not reverse video should be used. The default is False.

reverseWrap Specifies whether or not reverse-wraparound should be enabled. The default is False.

saveLines Specifies the number of lines in the terminal buffer beyond length of the window. The value consists of a number followed by an optional suffix. If no suffix is included, or the suffix is **l** (ell), the total length of the terminal buffer will be screens plus the length of the terminal window. If the suffix is **s** (ess), the total length of the terminal buffer will be (screens plus one) times the length of the terminal window. **dtterm** will try to maintain the same buffer-to-window ratio when the window is resized larger. The default is **4s**.

scrollBar Specifies whether or not the scrollbar should be visible. The default is True.

sunFunctionKeys Specifies whether or not Sun Function Key escape codes should be generated for function keys instead of standard VT220 escape sequences. The default is False.

termId Supplies the name used to select the correct response to terminal ID queries. Valid values are vt100, vt101, vt102, and vt220. The default is vt220.

termName Defines the name for the $TERM environment variable. The default is vt220.

title Specifies the window title. If the -e flag is used, the default will be the last component of the program's path. If the -e flag is not used, the default will be the last component of the name used to run **dtterm** (i.e., argv[0]).

ttyModes Specifies a string containing terminal-setting keywords and the characters to which they may be bound. Allowable keywords include: intr, quit, erase, kill, eof, eol, swtch, start, stop, brk, susp, dsusp, rprnt, flush, weras, and lnext. Keywords that do not apply to a specific architecture will be correctly parsed and ignored. Control characters may be specified as ^ followed by char (e.g. ^c or ^u), and ^? may be used to indicate delete. This is very useful for overriding the default terminal settings without having to do an **stty** every time a terminal process is started. The default is NULL.

userBoldFont Specifies an XFontSet to be used when displaying bold terminal text. It should be specified as a Motif XmFontList. Only character or mono spaced fonts are supported. The behavior when using proportional fonts is undefined. A default bold font will be generated based on the XLFD name of the userFont. If that font is not available, bold text will be generated by overstriking (with a one pixel offset) the userFont.

userFont Specifies an XFontSet to be used when displaying terminal text. It should be specified as a Motif XmFontList. Only character or mono spaced fonts are supported. The behavior when using proportional fonts is undefined. This font will not be used to display non-terminal text (menu bar, popup menu, dialog, etc.). The default is to use the XmNtextFontList value of the parent bulletin board (see XmBulletinBoard(3X)) in the same manner as the XmText widget.

visualBell Specifies that a visual bell is preferred over an audible one. Instead of ringing the terminal bell whenever a CTRL-G is received, the windows will be flashed. The default is False.

Pointer Usage

Note: **dtterm** allows you to select regions of text. Selection is based on the model specified in the Inter-Client Communication Conventions Manual (ICCCM). **dtterm** supports primary selection only. You can copy or paste selected text using primary transfer. Input is treated as keyboard input, and is inserted at the cursor. The select/insert operations and their default assignments are described below.

select The left button is used to select the text to be copied. Move the pointer to the beginning of the text to copy, press and hold the left button, move the cursor to the end of the text to copy, and release the button. Any currently selected text can be deselected by clicking the left button once without moving the mouse.

insert The middle button pastes the text from the primary selection, treating it as keyboard input.

Actions

bell *([Percentage])* This action rings the keyboard bell at the specified percentage above or below the base volume.

break ( ) This action send a break signal to the child process.

cancel ( ) This action sends a **CAN** (cancel) character to the child process.

do ( ) This action sends the escape sequence associated with the **Do** key to the child process.

edit-key *(string)* This action sends the escape sequence associated with the corresponding edit key to the child process. The interpretation of these keys is application specific. Valid values for string are find, insert, next, prior, remove, and select.

extend-start ( ) Start the extension of the currently selected text. extend-end ( )
Note: Extends the current selection. The amount of text selected depends on the number of mouse clicks.

function-key-execute *(num [,type])* This action sends the escape sequence associated with the corresponding function key *num* to the child process. Valid values for *num* are 1 through 35. If type is set to function (or not set at all), the escape sequence associated with function key *num* is sent to the child process. If *type* is set to **UDK**, then the string associated with user defined key *num* is sent to the child process.

grab-focus ( ) This action performs one of the following depending on the number of multiple mouse clicks. One click will deselect any selected text and set the selection anchor at the pointer position, two clicks will select a word, three clicks will select a line of text, and four clicks will select all text.

hard-reset ( ) This action will perform a hard reset on the terminal emulator.

help ( ) This action sends the escape sequence associated with the DEC VT220 Help key to the child process. The interpretation of this key is application specific.

keymap *(name)* This action dynamically defines a new translation table whose resource name is name with the suffix Keymap (case is significant). The name ″None″ restores the original translation table.

keypad-key-execute *(string)* This action sends the escape sequence associated with the corresponding keypad key to the child process. The interpretation of these keys are application specific. Valid values for *string* include: f1-f4, space, tab, enter, equal, multiply, add, separator, subtract, decimal, divide, and 0 - 9.

move-cursor *(direction)* This action sends the escape sequence associated with the corresponding cursor motion to the child process. The interpretation of these keys are application specific. Valid values for *direction* include: up, down, backward, and forward.

redraw-display ( ) This action redraws the contents of the text window.

scroll *(count [,units])* This action will scroll the display memory down if count is less than zero, or up if *count* is greater than zero. The number of lines scrolled is based on *count* and *units*. Valid values for *units* are page, halfpage, or line. The default for *units* is line.

select-adjust ( ) This action extends the selection. The amount of text selected depends on the number of mouse clicks:

1 click = char

2 clicks = word

3 clicks = line

4 clicks = buffer

select-all ( ) This action selects all text.

select-page ( ) This action selects all text on the screen.

self-insert ( ) This action sends the character associated with the key pressed to the child process.

soft-reset ( ) This action perform a soft reset of the terminal.

stop *(state)* This action either toggles, starts, or stops the process of reading data from the child process. Valid values for *state* are toggle, on, and off.

string *(string)* This action inserts the specified text *string* as if it had been typed. The *string* must be quoted if it contains whitespace or non-alphanumeric characters. The *string* is interpreted as a hex character constant if it begins with the characters 0x.

tab ( ) This action sends a tab to the child process.

visual-bell ( ) This action flashes the window quickly.

Virtual Bindings The bindings for virtual keys are vendor specific. Virtual bindings do not apply when the **dtterm** widget has input focus. For information about bindings for virtual buttons and keys, see VirtualBindings.

Files

/usr/bin/diff Contains the **diff** command.

Related Information

Files in *Operating system and device management* introduces you to files and the way you can work with them.

Input and output redirection in *Operating system and device management* describes how the operating system processes input and output.

---

du Command

Purpose

Summarizes disk usage.

Syntax

du [**-a** | **-s**] [**-k**] [**-m**] [**-g**][**-l**] [**-r**] [**-x**] [**-H** | **-L**][*File ...*]

Description

The **du** command displays the number of blocks used for files. If the *File* parameter specified is actually a directory, all files within the directory are reported on. If no *File* parameter is provided, the **du** command uses the files in the current directory.

If the *File* parameter is a directory, then the number of blocks reported is the sum of blocks allocated for the files in the directory and the blocks allocated for the directory itself.

If the object of the **du** command is a file or directory that exists inside a JFS2 snapshot, the **du** command gives information for the point-in-time object when the snapshot is created. This information does not include how much space is recovered if the snapshot itself is deleted.

Specifying the **-a** flag reports the number of blocks in individual files. Whether the **-a** flag is used or not, individual files specified by the *File* parameter are always listed.

Specifying the **-s** flag reports the total blocks for all specified files or all files in a directory.

The block count includes indirect blocks of each file. Block count is calculated in 512-byte units independent of the cluster size used by the system. Specifying the **-k** flag calculates the block count in 1024-byte units.

Notes:

1. Files with multiple links are counted and written for only one entry.
2. Block counts are based only on file size; therefore, unallocated blocks are not accounted for in the reported block counts.
3. If **du** cannot obtain the file attributes or cannot read directories, it reports an error and the exit status of the command is affected.

Flags

-a	For each file specified, displays the disk usage of the file. For each directory specified, displays the disk usage of each individual file within the directory, including all subdirectories. Contrast this flag with the **-s** flag.
-g	Calculates the block count in GB units rather than the default 512-byte units. The output values for the disk usage would be in floating point numbers as value of each unit in bytes is significantly high.
-H	If a symbolic link is specified on the command line, the **du** command shall count the size of the file or file hierarchy referenced by the link.
-k	Calculates the block count in 1024-byte units rather than the default 512-byte units.

-l Allocates blocks evenly among the links for files with multiple links. By default, a file with two or more links is counted only once.

-L If a symbolic link is specified on the command line or encountered during the traversal of a file hierarchy, the **du** command shall count the size of the file or file hierarchy referenced by the link.

-m Calculates the block count in MB units rather than the default 512-byte units. The output values for the disk usage would be in floating point numbers as value of each unit in bytes is significantly high.

-r Reports names of inaccessible files and directories. This is the default.

-s For each file specified, displays the disk usage of the file. For each directory specified, displays the total disk usage of all files within the directory, including all subdirectories. Contrast this flag with the **-a** flag.

-x When evaluating file sizes, evaluates only those files that reside on the same device as the file or directory specified by the *File* parameter. For example, you may specify a directory that contains files on several devices. In this case, the **-x** flag displays block sizes for all files that reside on the same device as the directory.

If all or any two of the **-k**, **-m** and **-g** flags are specified, the last one specified takes effect. The output of the disk usage with the flags **-m** and **-g** would be rounded off to the nearest second decimal digit.

Exit Status

This command returns the following exit values:

0 Successful completion.

>0 An error occurred.

Examples

1. To summarize the disk usage of a directory tree and each of its subtrees, enter:

   ```
   du /home/fran
   ```

   This displays the number of disk blocks in the `/home/fran` directory and each of its subdirectories.
2. To summarize the disk usage of a directory tree and each of its subtrees in 1024-byte blocks, enter:

   ```
   du -k /home/fran
   ```

   This displays the number of 1024-byte disk blocks in the **/home/fran** directory and each of its subdirectories.
3. To summarize the disk usage of a directory tree and each of its subtrees in MB blocks, enter:

   ```
   du -m /home/fran
   ```

   This displays the number of MB disk blocks rounded off to nearest 2nd decimal digit in the **/home/fran** directory and each of its subdirectories.
4. To summarize the disk usage of a directory tree and each of its subtrees in GB blocks, enter:

   ```
   du -g /home/fran
   ```

   This displays the number of GB disk blocks rounded off to nearest 2nd decimal digit in the **/home/fran** directory and each of its subdirectories.
5. To display the disk usage of each file, enter:

   ```
   du  -a /home/fran
   ```

   This displays the number of disk blocks contained in each file and subdirectory of the `/home/fran` directory. The number beside a directory is the disk usage of that directory tree. The number beside a regular file is the disk usage of that file alone.
6. To display only the total disk usage of a directory tree, enter:

   ```
   du  -s /home/fran
   ```

The **-s** flag instructs the **du** command to display only the sum total disk usage of the `/home/fran` directory and the files it contains. By default, the **du** command displays an error message if it cannot read a file or directory.

7. To display the disk usage of the files and file hierarchies referenced by all the symbolic links in addition to the normal files found during traversal of a the `/home/fran` directory, type:

   ```
   du -L /home/fran
   ```

8. To report the disk usage of the file or file hierarchy referenced by the symbolic link `mylink`, type:

   ```
   du -H mylink
   ```

Files

/usr/bin/du	Contains the **du** command.

Related Information

The **df** command.

The Directories in *Operating system and device management* explains working with directories and path names.

The Files in *Operating system and device management* provides information on working with files.

---

dump Command

Purpose

Dumps selected parts of an object file.

Syntax

dump { **-a -c -d -g -h -l -n -o -p-r -s -t -u -v -H -R -T**} [**-z***Name* [,*Number*] [**+z***Number*]] [**-t***Index* [**+t***Index*]] [**-X** {**32**|**64**|**32_64**|**d64**|**any**}] *File* ...

Note: Do not put a space between the **-z** *Name* flag and the ,*Number* parameter.

Description

The **dump** command dumps selected parts of the specified *File* parameter. The **dump** command accepts object files, archive object files, and executable files.

Flags

-a	Dumps the archive header of each member of each specified archive.
-c	Dumps the string table.
-d	Dumps the raw data for each section.
-g	Dumps the global symbols in the archive symbol table.
-h	Dumps section headers.
-l	Dumps line number information.
-n	Dumps all loader section information.
-o	Dumps each optional header.
-p	Suppresses header printing.
-r	Dumps relocation information.
-s	Dumps the raw data for each selection.
-t	Dumps symbol table entries.

-t*Index* Dumps only the index symbol table entry specified with the *Index* parameter. Use the **-t** flag with the **+t** flag to specify a range of symbol table entries.

+t*Index* Dumps the symbol entry in the range that ends with the *Index* parameter. The range starts at the first symbol table entry or at the entry specified by the **-t** flag.

-u Underlines the name of the *File* parameter.

-v Dumps the information in symbolic representation rather than numeric. Any flag except the **-o** flag and **-s** flag can be used with the **-v** flag.

-z*Name*[,*Number*] Dumps line number entries for the *Name* parameter or a range of line number entries that starts at the specified number.

+z*Number* Dumps all line numbers up to the *Number* parameter.

-H Dumps the header of the loader section. The **-H** flag applies only to executable files.

-R Dumps the relocation entries for the leader section. The **-R** flag applies only to executable files.

-T Dumps the symbol table entries for the loader section. The **-T** flag applies only to executable files.

-X *mode* Specifies the type of object file **dump** should examine. The *mode* must be one of the following:

32 Processes only 32-bit object files.

64 Processes only 64-bit object files.

32_64 Processes both 32-bit and 64-bit object files.

d64 Examines discontinued 64-bit XCOFF files (magic number = U803XTOCMAGIC).

any Processes all of the supported object files.

The default is to process 32-bit object files (ignore 64-bit objects). The *mode* can also be set with the **OBJECT_MODE** environment variable. For example, **OBJECT_MODE=64** causes **dump** to process any 64-bit objects and ignore 32-bit objects. The **-X** flag overrides the **OBJECT_MODE** variable.

Examples

1. To dump the string table of the `a.out` file, enter:
   ```
   dump -c a.out
   ```
2. To dump the contents of an XCOFF data section to standard output, enter:
   ```
   dump -d a.out
   ```
3. To dump the object file headers, enter:
   ```
   dump -o a.out
   ```
4. To dump line number information for the `a.out` file, enter:
   ```
   dump -l a.out
   ```
5. To dump relocation information for the `a.out` file, enter:
   ```
   dump -r a.out
   ```
6. To dump the contents of the `a.out` object file text section, enter:
   ```
   dump -s a.out
   ```
7. To dump symbol table information for the `a.out` object file, enter:
   ```
   dump -t a.out
   ```
8. To print symbol table entries `20` to `31` without header information, enter:
   ```
   dump -p -t20 +t30 a.out
   ```
9. To dump the object file headers from only 64-bit objects in lib.a, enter:
   ```
   dump -X64 -o lib.a
   ```

Related Information

The **ar** command, **size** command.

The **a.out** file, **ar** file.

---

dumpcheck Command

Purpose

Checks to see that the dump device and copy directory are able to receive the system dump. An error is logged by default if there will likely be insufficient resources to accommodate the dump.

Syntax

/usr/lib/ras/dumpcheck [[**-l**] [**-p**] [**-t** *TimeParameters*] [**-P**]] | [**-r**]

Description

The **/usr/lib/ras/dumpcheck** command is used to check the disk resources used by the system dump. The command logs an error if either the largest dump device is too small to receive the dump or there is insufficient space in the copy directory when the dump is to paging space.

dumpcheck is normally run by cron at 3:00 pm local time each day. This can be varied using the **-r** flag to remove it from root's **crontab** or **-t** *TimeParameters* to change the time at which dumpcheck is executed. It may also be configured from SMIT. **dumpcheck** is automatically added to root's **crontab** when the service aids are installed.

For maximum effectiveness, **dumpcheck** should be run when the system is most heavily loaded. At such times, the system dump is most likely to be at its maximum size. Also, even with **dumpcheck** watching the dump size, it may still happen that the dump would not fit on the dump device or in the copy directory at the time it happens. This could occur if there is a peak in system load right at dump time.

The **dumpcheck** function is installed as part of the service aids file set, installed automatically.

Flags

-l	Logs any warnings to the error log. This is the default if no parameters are specified.
-p	Prints any warnings produced to stdout.
-P	Indicates that the changes are to be made permanently; that is, they apply to subsequent executions of the **dumpcheck** facility. The **-P** flag is unnecessary with the **-t** and **-r** flags. If the **-P** flag is specified, **dumpcheck** simply changes the **crontab** entry without performing any checks.
-r	Removes the **crontab** entry for this function, effectively unconfiguring it. This command is normally run by **cron**. The **-r** flag must be specified alone. It is not valid with any other flags.
-t *TimeParameters*	Changes the time when **dumpcheck** is executed. The *TimeParameters* flag must be enclosed within single or double quotes. It specifies the **crontab** time parameters, the first five parameters of a line in the **crontab** file. See the **crontab** command for the format of the time parameters. The **-t** flag is invalid with the **-r** flag. If the **-t** flag is specified, **dumpcheck** just changes the **crontab** entry without performing any checks.

Security

This command can only be executed by the root user.

Examples

1. To check dump resources and have the results printed to standard output rather than logged, type:
   ```
   /usr/lib/ras/dumpcheck -p
   ```
   To make this change permanently; that is, to have it made in the **crontab** entry, type:
   ```
   /usr/lib/ras/dumpcheck -p -P
   ```
2. To have **dumpcheck** run at 9:00 am and 3:00 pm Monday through Friday, type:
   ```
   /usr/lib/ras/dumpcheck -t "0 9,15 * * 1-5"
   ```
   To return to the default, type:
   ```
   /usr/lib/ras/dumpcheck -t "0 15 * * *"
   ```
   You may also use SMIT to configure the times when **dumpcheck** executes.
3. To discontinue running this feature, type:
   ```
   /usr/lib/ras/dumpcheck -r
   ```
   You may also use SMIT for this task.

Related Information

The **sysdumpdev** command.

System Dump Facility in *AIX Version 6.1 Kernel Extensions and Device Support Programming Concepts*.

---

dumpctrl Command

Purpose

Manages system dumps and live dumps.

Syntax

dumpctrl -k

dumpctrl -R [**l** | **s**] [**-P**]

dumpctrl -s [**-c** | **-C** *comp-path-list*] [**-l** | **-L** *comp-alias-list*] [**-t** | **-T** *type_subtype*] [**-r**] [**-u**]

dumpctrl -qc [**-c** *comp-path-list*] [**-l** *comp-alias-list*] [**-t** *type_subtype*] [**-r**] [**-u**] [**-p** | **-P**]

dumcptrl -ql [**-p** | **-P**]

dumpctrl -qs [**-p** | **-P**]

dumpctrl [**-P**] [*global_attribute*]

dumpctrl [**-c** *comp-path-list*] [**l** *comp-alias-list*] [**-t** *type_subtype*] [**-r**] [**-u**] [**-n** | **-p** | **-P** | **-x**] [*per-component_attribute*]

Description

There are two types of dump components:

component
: Refers to a component specified with the RAS infrastructure (one created with the **ras_register()** kernel service).

legacy component
: Refers to a dump component specified with either the **dmp_add()** or the **dmp_ctl()** kernel service.

The **dumpctrl** command is used to obtain information about which components are registered for live dumps or system dumps, and to query and change dump characteristics.

Components are specified with the full path name, device logical alias, type or subtype. You can use multiple flags to specify multiple components or component lists.

Flags

At least one flag must be given.

-c *comp-path-list*
Specifies components by path name. Wildcards are allowed. Use **-c all** to specify all of the components.

-k
Refreshes the list of dumps of the kernel. This is run every 5 minutes by default. This period can be changed by editing the crontab for the root user and changing the entry for **/usr/sbin/dumpctrl -k**. For more information, see the **crontab** command. You need to run the **dumpctrl -k** command after adding or removing dumps by hand.

If the system is holding any dumps in the heap that it previously could not write to the file system, the system attempts to write those dumps and reclaim their storage space at this time.

-l *comp-alias-list*
Specifies components by alias. Wildcards are allowed.

-r
Dumps any subcomponents of the specified components.

-q *<cmd>*
Queries attributes for the live dump or system dump.

- The **-qc** flag shows component-specific live dump attributes and system dump attributes. The **-qc** flag can be used with the **-p** or **-P** flag to query persistent per-component attributes. The **-qc** flag shows the attributes for all components if neither the **-c**, **-l**, nor **-t** flag is given. In other words, **-c all** is the default.

  The following is a sample output for this command:

```
dumpctrl -qc -r -l vmm -l proc

-----------------------------------------------------------------------
     Component name                        | Have | Live Dump | System Dump
                                           |Alias |  /level   |  /level
-----------------------------------------------------------------------
vmm                                        | no   |  on/3     |  on/3
     .pft                                  | no   |  on/3     |  on/3
  ...
proc                                       | no   |  on/4     |  on/3
     ...
```

- **-ql** lists global live dump settings. **-ql** can be used with the **-p** or **-P** flag to query persistent global live dump settings.

-qs *type* shows global system dump attributes. **-qs** can be used with the **-p** or **-P** flag to query global system dump attributes.

-r
Includes components below the specified components in the component hierarchy.

-Rx
Restores dump settings to their defaults. *x* can be ″l″ for live dump settings, or ″s″ for system dump settings. This only resets the global dump settings. Individual components cannot be specified. The **-P** flag and a bosboot are required to ensure all of these settings remain in effect across a reboot.

-t *type_subtype*
Specifies a component by *type_subtype* names.

-s
Lists the path names and titles of all live dumps in the dump repository. If components are specified with the **-c**, **-l**, or **-t** flag, the list of dumps shown contains only dumps with the specified components. If components are specified with the **-C**, **-L**, or **-T** flag, the list of dumps shown contains only dumps with the specified failing components.

-C *comp-path-list*
Specifies components by path name. Wildcards are allowed. The reserved name ″all″ is also allowed to indicate all components. The **-C** flag is only valid with the **-s** flag.

-L *comp-alias-list* Specifies components by alias. Wildcards are allowed. The **-L** flag is only valid with the **-s** flag.

-T *type_subtype* Specifies a component by *type_subtype* names. The **-T** flag is only valid with the **-s** flag.

-u Includes components above the specified components in the component hierarchy.

Persistence flags

-p Changes apply only to newly created components, which are RAS infrastructure components created after the **dumpctrl** command runs.

-P Makes the specified changes permanent. Any changes made remain in effect across a reboot. If a bosboot is required, a message is produced to this effect. The **-P** flag applies to component attributes, the global enabling or disabling of live dump, the global live dump level, the enabling or disabling of legacy components, and the system dump device specifications.

-n Changes apply to existing components. The **-n** flag is the default if neither **-p** nor **-P** is specified. To apply changes to both current and newly created components, use the **-n** and **-p** flags.

-x Deletes this persistence specification. The **-x** flag deletes a permanent (**-P**) persistence specification. Note that the specification must be specified in the same manner as it was originally specified with the **-P** flag.

For more information about how the various dump attributes interact with persistence, see the live dump and system dump attribute tables in “Attributes.”

Attributes

The dump attributes can take the form `attribute=value`. For example,

```
dumpctrl dir=/usr/dumps freespc=20
```

This example sets dump directory to **/usr/dumps**, and the free space threshold to 20%.

Some shortcuts are provided, such as the **ldmpon** attribute, which is the same as **ldmpenable=yes**.

If components are given, unrecognized attributes are passed to callbacks of those components using **RASCD_DMP_PASS_THROUGH**.

The following table lists live dump attributes.

Table 1. Live dump attributes and defaults

Attribute	Specification	Default value
ldmpenable	Specifies whether live dump is enabled. The possible values are yes and no. You can use the **ldmpon** attribute instead of **ldmpenable=yes**, and the **ldmpoff** attribute instead of **ldmpenable=no**.	yes See the following note 1 on page 266 for more information.
dir	Specifies a live dump directory name.	/var/adm/ras/livedump
freespc	Specifies live dump free space threshold using a decimal value from 0 to 99.	25 (means 25%)
ldmplevel	Specifies the live dump level using a decimal value from 0 to 9. You can specify the **ldmpminimal**, **ldmpnormal**, or **ldmpdetail** attribute instead of **ldmplevel=1, 3, 7**	3 (normal) See the following note 1 on page 266 for more information.

Table 1. Live dump attributes and defaults (continued)

Attribute	Specification	Default value
heapsz	Specifies live dump heap size using a decimal value in megabytes.	0 See the following note 2 on page 266 for more information.
duptype	Specifies duplicate dump suppression type. The following are the possible values: • all • pre • post • none	all
maxfreeze	Specifies the maximum recommended system freeze interval using a decimal number in milliseconds.	100 ms

The following table lists system dump attributes.

Table 2. System dump attributes and defaults

Attribute	Specification	Default value
sdmpenable	Specifies whether system dump is enabled. The possible values are yes and no. You can also specify the **sdmpon** or **sdmpoff** instead of **sdmpenable=yes** or **sdmpenable=no**.	yes See the following note 3 on page 266 for more information.
legacyenable	Specifies whether dump legacy components are enabled. The possible values are yes and no. You can also specify the **legacyon** or **legacyoff** instead of **legacyenable=yes** or **legacyenable=no**.	yes
sdmplevel	Specifies the system dump level using a decimal value from 0 to 9. You can specify the **sdmpminimal**, **sdmpnormal**, or **sdmpdetail** attribute instead of **sdmplevel=1, 3, 7**	3 (normal) See the following note 4 on page 266 for more information.
copydir	Specifies a copy directory path name.	/var/adm/ras
forcecopy	Specifies whether the **forcecopy** attribute is enabled. The possible values are yes and no. If a dump must be copied from paging space at boot time, and there is not enough space in the copy directory, you are prompted to copy the dump to removable media if the **forcecopy** value is yes. If the value is no, the dump is not copied and the system boots normally, although the dump might be lost.	yes
keyseq	Specifies whether the key sequences always cause a dump. The possible values are yes and no.	no
primary	Specifies the primary dump device path name.	/dev/hd6 or /dev/lg_dumplv
secondary	Specifies the secondary dump device path name.	/dev/sysdumpnull

Notes:

1. The **ldmpenable** and **ldmplevel** attributes can be specified with or without components. If specified without components, the attributes apply to the corresponding global attributes.
2. The **heapsz** attribute (heap size) can be set to 0, meaning that, at dump initialization time, the system calculates the live dump heap size based on the amount of real memory, which is the minimum of 64MB and 1/64 the size of real memory.
3. Individual components must be specified when the **sdmpenable** attribute is given. If no components are given, the **sdmpenable** attribute cannot be specified, because the system dump cannot be totally disabled.
4. The **sdmplevel** attribute can be specified with or without components. If specified without components, it applies to the system default level.

The following table lists live dump attributes and their persistence.

Table 3. Live dump attributes and persistence

Attribute	Description	Persistence
ldmpenable	live dump enabled	Controlled by persistence flags, bosboot required with the **-P** flag.
dir	live dump directory	Takes effect immediately and upon system reboot.
freespc	live dump free space threshold	Takes effect immediately and upon system reboot.
ldmplevel	live dump level	Controlled by persistence flags, bosboot required with the **-P** flag.
heapsz	live dump heap size	Takes effect immediately and upon system reboot.
duptype	duplicate dump suppression type	Takes effect immediately and upon system reboot.
maxfreeze	maximum recommended system freeze interval	Takes effect immediately and upon system reboot.

Note: Persistence affects the attributes only when they apply to RAS infrastructure components. Persistence also controls the global live dump level and global enabled or disabled status.

The following table lists system dump attributes and their persistence.

Table 4. System dump attributes and persistence

Attribute	Description	Persistence
sdmpenable	system dump enabled	Controlled by persistence flags, bosboot required with the **-P** flag.
legacyenable	dump legacy components	Takes effect immediately, and upon system reboot with the **-P** flag. No bosboot required with the **-P** flag.
sdmplevel	system dump level	Controlled by persistence flags, bosboot required with the **-P** flag.
copydir	copy directory	Takes effect immediately and upon system reboot.
forcecopy	brings up the boot time menu if cannot copy	Takes effect immediately and upon system reboot.
keyseq	key sequences always cause a dump	Takes effect immediately and upon system reboot.

Table 4. System dump attributes and persistence (continued)

Attribute	Description	Persistence
primary	the primary dump device	Takes effect immediately, and upon system reboot with the **-P** flag. No bosboot required with the **-P** flag.
secondary	the secondary dump device	Takes effect immediately, and upon system reboot with the **-P** flag. No bosboot required with the **-P** flag.

Note: Persistence affects the attributes when they apply to components.

The **copydir, forcecopy, keyseq, primary**, and **secondary** attributes behave like their **sysdumpdev** command counterparts specified with the **sysdumpdev -d, -D, -k/-K, -p** and **-s** flags. For more information, see the **sysdumpdev** command in *AIX Version 6.1 Commands Reference, Volume 5*.

Exit Status

This command returns the following exit values:

0 Successful completion.

non-zero An error occurred. This command fails under the following conditions:

- One or more parameters are invalid.
- One or more attributes are invalid.
- A component cannot be specified.
- At least one component must be specified.
- The persistent specification cannot be found. (This can occur with the **-x** flag.)

Security

Only the root user can use this command.

Related information

The **livedumpstart** command in *AIX Version 6.1 Commands Reference, Volume 3*.

---

dumpfs Command

Purpose

Dumps file system information.

Syntax

dumpfs { *FileSystem* | *Device* }

Description

The **dumpfs** command prints out the superblock, i-node map, and disk map information for the file system or special device specified. This listing is used to find out file system information. Primarily, the **dumpfs** command is for debugging purposes.

The **dumpfs** command can also run against a JFS2 snapshot. The **dumpfs** command prints out the superblock, snapshot map, and block map xtree copy for the specified snapshot.

Note: The **dumpfs** command will not work on UDF, NFS, or JFS diskettes.

Security

Attention RBAC users and Trusted AIX users: This command can perform privileged operations. Only privileged users can run privileged operations. For more information about authorizations and privileges, see Privileged Command Database in *Security*. For a list of privileges and the authorizations associated with this command, see the **/etc/security/privcmds** database file.

Examples

To print the information for **/dev/hd4**, enter:

```
dumpfs /dev/hd4
```

Related Information

The **fsck** command, **mkfs** command.

---

echo Command

Purpose

Writes character strings to standard output.

Syntax

echo [*String ...*]

Description

The **echo** command writes character strings to standard output. *String*s are separated by spaces, and a new-line character follows the last *String* parameter specified. If no *String* parameter is specified, a blank line (new-line character) is displayed.

Normally you could distinguish between a flag and a string that begins with a hyphen by using a — (double hyphen). Since no flags are supported with the **echo** command, a — (double hyphen) is treated literally.

The **echo** command recognizes the following escape conventions:

\a	Displays an alert character.
\b	Displays a backspace character.
\c	Suppresses the new-line character that otherwise follows the final argument in the output. All characters following the **\c** sequence are ignored.
\f	Displays a form-feed character.
\n	Displays a new-line character.
\r	Displays a carriage return character.
\t	Displays a tab character.
\v	Displays a vertical tab character.
****	Displays a backslash character.
\0*Number*	Displays an 8-bit character whose ASCII value is a 0-, 1-, 2-, or 3-digit octal number.

> **Note:** The **bsh**, **ksh**, and **csh** commands each contain a built-in **echo** subcommand. The **echo** command and the **bsh** and **ksh echo** subcommands work the same way. The **csh echo** subcommand does not work the same way as the **echo** command. For information on the **echo** subcommands, see ″Bourne shell built-in commands,″ ″Regular built-in command descriptions for the Korn shell or POSIX shell,″ and ″C shell built-in commands″ in *Operating system and device management*.

The \ (backslash) is a quote character in the shell. This means that unless the \ is used with an escape character or enclosed in quotes, for example ″\″ or '\', the shell removes the backslashes when the command is expanded.

After shell expansion, the **echo** command writes the output based on the escape sequences in the input. Refer to the Backslash Reduction table for an example comparison of how backslashes in a command are first reduced by the shell and then by the **echo** command:

Backslash Reduction		
Command Entered	**After Shell Expansion**	**After echo Command Processing**
echo hi\\\\there	echo hi\\there	hi\there
echo 'hi\\\\there'	echo 'hi\\\\there'	hi\\there
echo ″hi\\\\there'	echo ″hi\\there″	hi\there

Exit Status

This command returns the following exit values:

0 Successful completion.
>0 An error occurred.

Examples

1. To write a message to standard output, enter:

   ```
   echo Please insert diskette . . .
   ```

2. To display a message containing special characters, enter:

   ```
   echo "\n\n\nI'm at lunch.\nI'll be back at 1:00."
   ```

   This skips three lines and displays the message:

   ```
   I'm at lunch.
   I'll be back at 1:00.
   ```

   Note: You must put the message in quotation marks if it contains escape sequences. Otherwise, the shell interprets the \ (backslash) as a metacharacter and treats the \ differently.

3. To use the **echo** command with pattern-matching characters, enter:

   ```
   echo The back-up files are: *.bak
   ```

   This usage displays the message `The back-up files are:` followed by the file names in the current directory ending with `.bak`.

4. To add a single line of text to a file, enter:

   ```
   echo Remember to set the shell search path to $PATH. >>notes
   ```

   This usage adds the message to the end of the file notes after the shell substitutes the value of the **PATH** shell variable.

5. To write a message to the standard error output, enter:

   ```
   echo Error: file already exists. >&2
   ```

   This command redirects the error message to standard error. If the `>&2` is omitted, the message is written to standard output.

File

/usr/bin/echo	Contains the **echo** command.

Related Information

The **bsh** command, **csh** command, **ksh** command, **printf** command.

Input and output redirection in *Operating system and device management* describes how the operating system processes input and output and how to use the redirect and pipe symbols.

The Shells in *Operating system and device management* describes what shells are, the different types of shells, and how shells affect the way commands are interpreted.

---

ed or red Command

Purpose

Line editor for text files.

Syntax

ed [**-p** *String*] [**-s** | **-**] [*File*]

red [**-p***String*] [**-s** | **-**] [*File*]

Description

The **ed** command starts the ed editor line-editing program. The ed editor works on only one file at a time by copying it into a temporary edit buffer and making changes to that copy. The ed editor is part of a family of editors that also includes the edit editor, ex editor, and vi editor. The ed editor makes the changes you specify in a buffer. It does not alter the file itself until you use the write (**w**) subcommand.

You can specify the name of the file you want to edit when you start the ed editor with the **ed** command, or you can use the **e** subcommand. When the **ed** command reads a new file into the buffer, the contents of that file replace the buffer's previous contents.

The **red** command is a restricted version of the **ed** command, for use with the restricted shell (**rsh**). With the **red** command, you edit only files that reside in the current directory or in the **/tmp** directory; you cannot use the **!** subcommand.

An ed editor subcommand consists of zero, one, or two addresses, followed by a single-character subcommand, followed by optional parameters to that subcommand. The addresses specify one or more lines in the buffer. Because every subcommand has default addresses, it is frequently unnecessary to specify addresses.

The ed editor allows editing only the current line unless you address another line in the buffer. You can move and copy only complete lines of data. The ed editor is useful for editing large files or for editing within a shell program.

The ed editor operates in one of two modes:

command mode	In command mode, the ed editor recognizes and runs subcommands. When you start the ed editor, it is in command mode. Type a **.** (period) and press Enter to confirm that you are in command mode.

text input mode In text input mode, the ed editor allows you to enter text into the file buffer but does not recognize subcommands. You enter text input mode by using the **a**subcommand, **c** subcommand, or **i** subcommand. You exit text input mode and return to the command mode by typing a **.** (period) alone at the beginning of a line. To place a **.** (period) into the buffer while in text input mode, enter a character followed by the **.** (period). Then, exit text input mode and use the **s** subcommand to remove the character.

The following list provides the maximum limits of the **ed** editor.

- 64 characters per file name
- 256 characters per global subcommand list
- 128,000 character buffer size

Note: The buffer contains the original file as well as editing information.

The maximum number of lines depends on the amount of memory available. The maximum file size depends on the amount of physical data storage (disk or tape drive) available or on the maximum number of lines permitted in user memory.

Flags

-p *String* Sets the editor prompt to the *String* parameter. The default for *String* is a null value (no prompt).

-s Suppresses character counts that the editor displays with the **e** subcommand, **r** subcommand, and **w** subcommand. This flag also suppresses diagnostic messages for the **e** subcommand and the **q** subcommand, and suppresses the ! (exclamation point) prompt after an **!** subcommand.

- Provides the same functions as the **-s** flag.

Pattern Matching

The ed editor supports a limited form of special pattern-matching characters that you can use as regular expressions (REs) to construct pattern strings. You can use these patterns in addresses to specify lines and in some subcommands to specify portions of a line.

Regular Expressions

The following REs match a single character or a collating element as follows:

Character Matches itself and can be any ordinary character (other than one of the special pattern-matching symbols).

. Matches any single character except the new-line character.

[*String*] Matches any one character in the string. Certain pattern-matching characters have special meanings within brackets as follows:

^ Matches any character except the characters in the *String* parameter and the new-line character if the first character of the *String* parameter is a ^ (circumflex). This condition is true only if the ^ is the first character in the string, [^*String*].

- Indicates a range of consecutive ASCII characters according to the current collating sequence. For example, [a-f] can be equivalent to [abcdef] or [aAbBcCdDeEfF] or [abcdef] and could even include accented a and e characters. A collating sequence can define equivalence classes for characters.

The minus sign loses its significance if it occurs as the first character in the string, [-*String*]; if it immediately follows an initial circumflex, [^-*String*]; or if it appears as the last character in the string, [*String*-].

] Functions as a part of the string rather than as the string terminator, when the] (right bracket) is the first character in the string, []*String*], or when it immediately follows an initial circumflex, [^]*String*].

Forming Patterns

The following rules describe how to form patterns from REs:

- An RE that consists of a single, ordinary character matches that same character in a string.
- An RE followed by an * (asterisk) matches zero or more occurrences of the character that the RE matches. For example, the following pattern:

  ```
  ab*cd
  ```

  matches each of the following strings:

  ```
  acd
  abcd
  abbcd
  abbbcd
  ```

  but not the following string:

  ```
  abd
  ```

  If a choice exists, the longest matching leftmost string is chosen. For example, given the following string:

  ```
  122333444
  ```

  the pattern .* matches 122333444, the pattern .*3 matches 122333, and the pattern .*2 matches 122.
- An RE followed by:

\{*m*\}	Matches *exactly m* occurrences of the character matched by the RE.
\{*m*,\}	Matches *at least m* occurrences of the character matched by the RE.
\{*m*,*n*\}	Matches *any number* of occurrences of the character matched by the RE from *m* to *n* inclusive.

  The numbers *m* and *n* must be integers from 0 to 255, inclusive. Whenever a choice exists, this pattern matches as many occurrences as possible.
- You can combine REs into patterns that match strings containing that same sequence of characters. For example, the pattern AB*CD matches the string AB*CD, and the pattern [A-Za-z]*[0-9]* matches any string that contains any combination of alphabetic characters (including none), followed by any combination of numerals (including none).
- The character sequence \(*Pattern*\) marks a subpattern that matches the same string the sequence would match if it were not enclosed.
- The characters *Number* match the same string of characters that a subpattern matched earlier in the pattern (see the preceding rule). The pattern of the *Number* parameter represents a digit. The pattern *Number* matches the string matched by the occurrence of the subpattern specified by the *Number* parameter, counting from left to right.

  For example, the following pattern:

  ```
  \(A\)\(B\)C\2\1
  ```

  matches the string ABCBA. You can nest subpatterns.

Restricting What Patterns Match

You can restrict a pattern to match only the first segment of a line, the final segment, or the entire line. The null pattern, // (two slashes), duplicates the previous pattern.

Matching the First Segment of a Line: The ^*Pattern* parameter matches only a string that begins in the first character position on a line.

Matching the Last Segment of a Line: The *Pattern*$ parameter matches only a string that ends with the last character (not including the new-line character) on a line.

Matching the Entire Line: The ^*Pattern*$ parameter restricts the pattern to match an entire line.

Addressing Lines

The ed editor uses three types of addresses: line number addresses, addresses relative to the current line, and pattern addresses. The current line (usually the last line affected by a subcommand) is the point of reference in the buffer.

You can use line addressing to do the following:

- Designate a new current line
- Display the addressed line or lines
- Cause a command to act on a certain line or lines

Subcommands that do not accept addresses regard the presence of an address as an error. Subcommands that accept addresses can use either given or default addresses. When given more addresses than it accepts, a command uses the last (rightmost) ones.

In most cases, commas (,) separate addresses (for example `2,8`). Semicolons (;) also can separate addresses. A semicolon between addresses causes the ed editor to set the current line to the first address and then calculate the second address (for example, to set the starting line for a search). In a pair of addresses, the first address must be numerically smaller than the second.

You can use line numbers and symbolic addresses to perform the following tasks:

- Addressing the current line
- Addressing a line by number
- Addressing the line before the first line
- Addressing the last line
- Addressing a line above an addressed line
- Addressing a line below an addressed line
- Addressing the first line through the last line
- Addressing the current line through the last line
- Addressing a group of lines
- Addressing the next line that contains a specified pattern
- Addressing the previous line that contains a specified pattern
- Addressing a marked line

Addressing the Current Line

A **.** (period) addresses the current line. The . (period) is the default for most ed editor subcommands and does not need to be specified.

Addressing a Line by Number

To address a specified line of the buffer, type:

```
Number
```

where the *Number* parameter represents a line number. For example:

```
2253
```

addresses line number 2253 as the current line.

Addressing the Line before the First Line

To address the line before the first line of the buffer, type:

```
0
```

Addressing the Last Line

To address the last line of the buffer, type:

```
$
```

Addressing a Line above an Addressed Line

To specify an address that is a specified number of lines above the current line, type:

```
-Number
```

where the *Number* parameter is the specified number of lines above the current line that you want to address. For example:

```
-5
```

addresses the line five lines above the current line as the current line.

You also can specify only a - to address the line immediately above the current line. The minus sign has a cumulative effect. For example, the address - - (two minus signs) addresses the line two lines above the current line.

Addressing a Line below an Addressed Line

To specify an address that is a specified number of lines below the current line, type:

```
+Number
```

where the *Number* parameter is the specified number of lines below the current line that you want to address. The + (plus sign) is optional. For example:

```
+11
```

addresses the line 11 lines below the current line as the current line.

You also can specify only a + to address the line immediately below the current line. The + has a cumulative effect. For example, the address + + (two plus signs) addresses the line two lines below the current line.

Addressing the First Line through the Last Line

To address the first line through the last line, type:

```
,
```

The , (comma) represents the address pair 1,$ (first line through last line). The first line becomes the current line.

Addressing the Current Line through the Last Line

To address the current line through the last line, type:

```
;
```

The ; (semicolon) represents the address pair .,$ (current line through last line).

Addressing a Group of Lines

To address a group of lines, type:

```
FirstAddress,LastAddress
```

where the *FirstAddress* parameter is the line number (or symbolic address) of the first line in the group you want to address, and the *LastAddress* parameter is the line number (or symbolic address) of the last line in the group. The first line in the group becomes the current line. For example:

```
3421,4456
```

addresses the lines 3421 through 4456. Line 3421 becomes the current line.

Addressing the Next Line That Contains a Specified Pattern

To address the next line that contains a matching string, type:

```
/Pattern/
```

where the *Pattern* parameter is a character string or regular expression. The search begins with the line after the current line and stops when it finds a match for the pattern. If necessary, the search moves to the end of the buffer, wraps around to the beginning of the buffer, and continues until it either finds a match or returns to the current line. For example:

```
/Austin, Texas/
```

addresses the next line that contains Austin, Texas as the current line.

Addressing the Previous Line That Contains a Specified Pattern

To address the previous line that contains a match for the pattern, type:

```
?Pattern?
```

where the *Pattern* parameter is a character string or regular expression. The ?*Pattern*? construction, like /*Pattern*/, can search the entire buffer, but it searches in the opposite direction. For example:

```
?Austin, Texas?
```

addresses the previous line that contains Austin, Texas as the current line.

Addressing a Marked Line

To address a marked line with the **k** subcommand, type:

```
'x
```

where the *x* parameter is a lowercase letter *a* to *z*. For example:

```
'c
```

addresses the line marked as c with the **k** subcommand.

Subcommands

Use the ed editor subcommands to perform the following actions:

- Editing a file
- Manipulating files
- Performing miscellaneous functions
  - Changing the prompt string
  - Entering system commands
  - Exiting the ed editor
  - Requesting help

In most cases, you can enter only one ed editor subcommand on a line. However, you can add the **l** (list) and **p** (print) subcommands to any subcommand except the **e** (edit), **E** (Edit), **f** (file), **q** (quit), **Q** (Quit), **r** (read), **w** (write), and **!** (operating system commands) subcommands.

The **e**, **f**, **r**, and **w** subcommands accept file names as parameters. The ed editor stores the last file name used with a subcommand as a default file name. The next **e**, **E**, **f**, **r**, or **w** subcommand given without a file name uses the default file name.

The ed editor responds to an error condition with one of two messages: ? (question mark) or `?File`. When the ed editor receives an Interrupt signal (the Ctrl-C key sequence), it displays a ? and returns to command mode. When the ed editor reads a file, it discards ASCII null characters and all characters after the last new-line character.

Editing a File

You can use the ed editor subcommands to perform the following tasks:

- Adding text
- Changing text
- Copying text
- Deleting text
- Displaying text
- Joining and splitting lines
- Making global changes
- Marking text
- Moving text
- Saving text
- Searching text
- Substituting text
- Undoing text changes

Note: In the following descriptions of ed editor subcommands, default addresses are shown in parentheses. Do not type the parentheses. The address . (period) refers to the current line. A . (period) in the first position of an otherwise empty line is the signal to return to command mode.

Adding Text

(.)**a** [**l**] [**n**] [**p**] *Text***.**

The **a** (append) subcommand adds text to the buffer *after* the addressed line. The **a** subcommand sets the current line to the last inserted line, or, if no lines were inserted, to the addressed line. A 0 address adds text to the beginning of the buffer.

Type the **l** (list), **n** (number), or **p** (print) optional subcommand if you want to display the added text.

Type your text, pressing the Enter key at the end of each line. If you do not press Enter at the end of each line, the ed editor automatically moves your cursor to the next line after you fill a line with characters. The ed editor treats everything you type before you press Enter as one line, regardless of how many lines it takes up on the screen.

Type a . (period) at the start of a new line, after you have typed all of your text.

(.)**i** [**l**] [**n**] [**p**]*Text***.**

The **i** (insert) subcommand inserts text *before* the addressed line and sets the current line to the last inserted line. If no lines are inserted, the **i** subcommand sets the current line to the addressed line. You cannot use a 0 address for this subcommand.

Type the **l** (list), **n** (number), or **p** (print) optional subcommand if you want to display the inserted text.

Type your text, pressing the Enter key at the end of each line. If you do not press Enter at the end of each line, the ed editor automatically moves your cursor to the next line after you fill a line with characters. The ed editor treats everything you type before you press Enter as one line, regardless of how many lines it takes up on the screen.

Type a . (period) at the start of a new line, after you have typed all of your text.
Note: The **i** subcommand differs from the **a** subcommand only in the placement of the text.

You can use different ed editor subcommands to add text in different locations. Use the preceding format to perform the following editing tasks:

- Adding text after the current line
- Adding text before the current line
- Adding text after an addressed line
- Adding text before an addressed line
- Adding text after lines that contain a search pattern
- Adding text before lines that contain a search pattern
- Adding text after lines that do not contain a search pattern
- Adding text before lines that do not contain a search pattern

To Add Text after the Current Line:

1. Type the following subcommand:
   ```
   a[l][n][p]
   ```
   where **l**, **n**, and **p** are optional subcommands that display the added text.
2. Type the text, and press Enter.
3. Type a **.** (period), and press Enter again to return to command mode.

To Add Text before the Current Line:

1. Type the following subcommand:
   ```
   i[l][n][p]
   ```
   where **l**, **n**, and **p** are optional subcommands that display the added text.
2. Type the text, and press Enter.
3. Type a **.** (period), and press Enter again to return to command mode.

To Add Text after an Addressed Line:

1. Type the following subcommand:
   ```
   Addressa[l][n][p]
   ```
   where the *Address* parameter is the line number of the line that the inserted text should follow. The **l**, **n**, and **p** optional subcommands display the added text.
2. Type the text, and press Enter.
3. Type a **.** (period), and press Enter again to return to command mode.

To Add Text before an Addressed Line:

1. Type the following subcommand:
   ```
   Addressi[l][n][p]
   ```
   where the *Address* parameter is the line number of the line that the inserted text should precede. The **l**, **n**, and **p** optional subcommands display the added text.
2. Type the text, and press Enter.
3. Type a **.** (period), and press Enter again to return to command mode.

To Add Text after Lines That Contain a Search Pattern:

1. Type the following subcommand:
   ```
   [Address]g/Pattern/a[l][n][p]
   ```

where *Address* is an optional parameter that specifies the range of lines to search for the pattern specified in the *Pattern* parameter. The *Pattern* parameter is a character string or regular expression. If you omit the *Address* parameter, the ed editor searches the entire file for lines that contain the pattern. The **l**, **n**, and **p** optional subcommands display the added text.

2. Type a backslash:

   \

3. Type the text. To start new lines within the added text, type a backslash:

   \

   and press Enter. The text you type is added after every line that contains the pattern specified in the command.

4. To return to command mode, press Enter.

To Add Text before Lines That Contain a Search Pattern:

1. Type the following subcommand:

   ```
   [Address]g/Pattern/i[l][n][p]
   ```

   where *Address* is an optional parameter that specifies the range of lines to search for the pattern specified in the *Pattern* parameter. The *Pattern* parameter is a character string or regular expression. If you omit the *Address* parameter, the ed editor searches the entire file for lines that contain the pattern. The **l**, **n**, and **p** optional subcommands display the added text.

2. Type a backslash:

   \

3. Type the text. To start new lines within the added text, type a backslash:

   \

   and press Enter. The text you type is added before every line that contains the pattern specified in the command.

4. To return to command mode, press Enter.

To Add Text after Lines That Do Not Contain a Search Pattern:

1. Type the following subcommand:

   ```
   [Address]g/Pattern/a[l][n][p]
   ```

   where *Address* is an optional parameter that specifies the range of lines to search for lines that do not contain the pattern specified in the *Pattern* parameter. The *Pattern* parameter is a character string or regular expression. If you omit the *Address*, the ed editor searches the entire file for lines that do not contain the pattern. The **l**, **n**, and **p** optional subcommands display the added text.

2. Type a backslash:

   \

3. Type the text. To start new lines within the added text, type a backslash:

   \

   and press Enter. The text you type is added after every line that does not contain the pattern specified in the command.

4. To return to command mode, press Enter.

To Add Text before Lines That Do Not Contain a Search Pattern:

1. Type the following subcommand:

   ```
   [Address]g/Pattern/i[l][n][p]
   ```

where *Address* is an optional parameter that specifies the range of lines to search for lines that do not contain the pattern specified in the *Pattern* parameter. The *Pattern* parameter is a character string or regular expression. If you omit the *Address* parameter, the ed editor searches the entire file for lines that do not contain the pattern. The **l**, **n**, and **p** optional subcommands display the added text.

2. Type a backslash:

   \

3. Type the text. To start new lines within the added text, type a backslash:

   \

   and press Enter. The text you type is added before every line that does not contain the pattern specified in the command.

4. To return to command mode, press Enter.

Changing Text

(.,.)**c** [**l**] [**n**] [**p**]*Text***.**

The **c** (change) subcommand deletes the addressed lines you want to replace and then replaces them with the new lines you enter. The **c** subcommand sets the current line to the last new line of input, or, if no input existed, to the first line that was not deleted.

Type the **l** (list), **n** (number), or **p** (print) optional subcommand if you want to display the inserted text.

Type the new text, and press Enter at the end of each line. When you have entered all of the new text, type a . (period) on a line by itself.

You can change text in several different ways with the ed editor. Use the preceding format to perform the following editing tasks:

- Changing the text of the current line
- Changing the text of a line or group of lines
- Changing text of lines that contain a specified pattern
- Changing text of lines that do not contain a specified pattern

To Change the Text of the Current LIne:

1. Type the following subcommand:

   ```
   c[l][n][p]
   ```

   where **l**, **n**, and **p** are optional subcommands that display the changed text.

2. Type the text, and press Enter.
3. Type a **.** (period), and press Enter again to return to command mode.

To Change the Text of a Line or Group of Lines:

1. Type the following subcommand:

   ```
   Addressc[l][n][p]
   ```

   where the *Address* parameter is the address of the line or group of lines to change. The **l**, **n**, and **p** optional subcommands display the changed text.

2. Type the text, and press Enter.
3. Type a **.** (period), and press Enter again to return to command mode.

To Change the Text of Lines That Contain a Specified Pattern:

1. Type the following subcommand:

   ```
   Addressg/Pattern/c[l][n][p]
   ```

where the *Address* parameter is the address of the group of lines that you want to search for the pattern specified with the *Pattern* parameter. The **l**, **n**, and **p** optional subcommands display the changed text.

2. Type a backslash:
   \
3. Type the new text. To start new lines within the new text, type a backslash:
   \

   and press Enter.
4. To return to command mode, press Enter again, type a . (period), and press Enter again.

To Change the Text of Lines That Do Not Contain a Specified Pattern:

1. Type the following subcommand:
   ```
   Addressv/Pattern/c[l][n][p]
   ```

   where the *Address* parameter is the address of the group of lines that you want to search for the pattern specified with the *Pattern* parameter. The **l**, **n**, and **p** optional subcommands display the changed text.
2. Type a backslash:
   \
3. Type the new text. To start new lines within the new text, type a backslash:
   \

   and press Enter.
4. To return to command mode, press Enter again, type a . (period), and press Enter again.

Copying Text

(.,.)**t***Address* [**p**] [**l**] [**n**]

The **t** (transfer) subcommand inserts a copy of the addressed lines after the line specified by the *Address* parameter. The **t** subcommand accepts the 0 address to insert lines at the beginning of the buffer.

The **t** subcommand sets the current line to the last line copied.

Type the **l** (list), **n** (number), or **p** (print) optional subcommand if you want to display the transferred text.

Copying a line or a set of lines leaves the specified lines in their original location and puts a copy in the new location. You can select the lines to copy by specifying an address or pattern. Use the preceding format to perform the following editing tasks:

- Copying the current line
- Copying lines specified by address
- Copying lines that contain a specified pattern
- Copying lines that do not contain a specified pattern

To Copy the Current Line:

1. Type the following subcommand:
   ```
   tAddress[l][n][p]
   ```

   where the *Address* parameter is the line number or symbolic address of the line you want a copy of the current line to follow. The **l**, **n**, and **p** optional subcommands display the copied line.
2. Type the text, and press Enter.

3. Type a **.** (period), and press Enter again to return to command mode.

To Copy Lines Specified by Address:

1. Type the following subcommand:

   ```
   LineNumbertDestinationAddress[l][n][p]
   ```

   where the *LineNumber* parameter is the address of the lines you want to copy, and the *DestinationAddress* parameter is the line you want the copy to follow. The **l**, **n**, and **p** optional subcommands display the copied line.
2. Type the text, and press Enter.
3. Type a **.** (period), and press Enter again to return to command mode.

To Copy Lines That Contain a Specified Pattern: Type the following subcommand:

```
[Address]g/Pattern/t[DestinationAddress][l][n][p]
```

where *Address* is an optional parameter that specifies the range of lines to search for lines that contain the specified pattern, the *Pattern* parameter is the text you are searching for, and the *DestinationAddress* is an optional parameter that identifies the line you want the copied text to follow. The **l**, **n**, and **p** optional subcommands display the copied line.

If you omit the *Address* parameter, the ed editor searches the entire file for lines that contain the pattern. If you omit the *DestinationAddress* parameter, the copied text is placed after the current line.

To Copy Lines That Do Not Contain a Specified Pattern: Type the following subcommand:

```
[Address]v/Pattern/t[DestinationAddress][l][n][p]
```

where *Address* is an optional parameter that specifies the range of lines to search for lines that do not contain the specified pattern, the *Pattern* parameter is the text, and the *DestinationAddress* is an optional parameter that identifies the line you want the copied text to follow. The **l**, **n**, and **p** optional subcommands display the copied line.

If you omit the *Address* parameter, the ed editor searches the entire file for lines that do not contain the pattern. If you omit the *DestinationAddress* parameter, the copied text is placed after the current line.

Deleting Text

(.,.)**d** [**l**] [**n**] [**p**]

The **d** (delete) subcommand removes the addressed lines from the buffer. The line after the last line deleted becomes the current line. If the deleted lines were originally at the end of the buffer, the new last line becomes the current line.

Type the **l** (list), **n** (number), or **p** (print) optional subcommand if you want to display the deletion.

The ed editor provides several ways to delete text. Use the preceding format to perform the following editing tasks:

- Deleting the current line
- Deleting a line or group of lines
- Deleting a line or group of lines that contain a specified pattern
- Deleting a line or group of lines that does not contain a specified pattern
- Deleting text from the current line
- Deleting text within selected lines
- Deleting text from addressed lines
- Deleting text from lines that contain a specified pattern

- Deleting a pattern from lines that contain a different specified pattern
- Deleting a pattern from lines that do not contain a different specified pattern

To Delete the Current Line: Type the following subcommand:

```
d[l][n][p]
```

where **l**, **n**, and **p** are optional subcommands that display the deleted line.

To Delete a Line or Group of Lines: Type the following subcommand:

```
Addressd[l][n][p]
```

where the *Address* parameter is the line number or symbolic address of the lines you want to delete, and **l**, **n**, and **p** are optional subcommands that display the deleted line or lines.

To Delete a Line or Group of Lines That Contain a Specified Pattern: Type the following subcommand:

```
[Address]g/Pattern/d[l][n][p]
```

where *Address* is an optional parameter that specifies the line number or symbolic address of the lines you want to search, and the *Pattern* parameter is a character string or regular expression that represents the text you want to find. If you omit the *Address* parameter, the ed editor searches the entire file for lines that contain the specified pattern. The **l**, **n**, and **p** optional subcommands display the deleted line or lines.

To Delete a Line or Group of Lines That Does Not Contain a Specified Pattern: Type the following subcommand:

```
[Address]v/Pattern/d[l][n][p]
```

where *Address* is an optional parameter that specifies the line number or symbolic address of the lines you want to search, and the *Pattern* parameter is a character string or regular expression that represents the text you want to find. If you omit the *Address* parameter, the ed editor searches the entire file for lines that do not contain the specified pattern. The **l**, **n**, and **p** optional subcommands display the deleted line or lines.

To Delete Text from the Current Line:

1. Type the following subcommand:

   ```
   s/Pattern
   ```

   where the *Pattern* parameter is a character string or regular expression that represents the text you want to delete.
2. To delete the *first instance* of the pattern from the line, type:

   ```
   //
   ```

   OR

   To delete *every instance* of the pattern from the line, type:

   ```
   //g
   ```
3. If you want to display the deletion, type one of the following optional subcommands:

   ```
   l

   n

   p
   ```
4. Press Enter.

To Delete Text within Selected Lines:

1. Type the address of a group of lines to select (or skip this step to select all lines).
2. To select the lines indicated by the *Pattern* parameter in step 4, type:

   ```
   g
   ```

   OR

   To select the lines *not* indicated by the *Pattern* parameter in step 4, type:

   ```
   v
   ```
3. To enter the text you want to search, type the following subcommand:

   ```
   /Pattern/s
   ```

   where the *Pattern* parameter is the text you want to search.
4. Type one of the following commands to make the desired deletion:

   To delete the first instance of the *Pattern* parameter within each selected line, type:

   ```
   ///
   ```

   To delete every instance of the *Pattern* parameter within each selected line, type:

   ```
   ///g
   ```

   To delete the first specified number of occurrences of the *Pattern* parameter on each selected line (where the *Number* parameter is an integer), type:

   ```
   ///Number
   ```

   To delete the first character string indicated by the *OtherPattern* parameter within each line selected by the *Pattern* parameter (where the *OtherPattern* parameter is the pattern you want to search), type:

   ```
   /OtherPattern//
   ```

   To delete every instance of the *OtherPattern* parameter within each line selected by the *Pattern* parameter, type:

   ```
   /OtherPattern//g
   ```

   To delete the first specified number of occurrences of the *OtherPattern* parameter on each line selected by the *Pattern* parameter (where the *Number* parameter is an integer), type:

   ```
   /OtherPattern//Number
   ```
5. If you want to display the deletion, type one of the following optional subcommands:

   ```
   l
   ```

   ```
   n
   ```

   ```
   p
   ```
6. Press Enter.

For example, to delete all instances of a pattern from *a range of lines*, type:

```
38,$g/tmp/s/gn
```

The previous example searches all the lines from line 38 to the last line (`38,$`) for the `tmp` character string and deletes every instance (`/g`) of that character string within those lines. It then displays the lines that had text deleted from them and their line numbers (`n`).

To delete all instances of a pattern from *all lines* that contain that pattern, type:

```
g/rem/s///gl
```

The previous example searches the entire file (address parameter is omitted) for all lines that contain (g) the rem character string. It deletes all instances (///g) of the rem character string from each of those lines and then displays the lines that had text deleted from them, including the nonprinting characters in those lines (l).

To Delete Text from Addressed Lines:

1. Type the following subcommand:

   ```
   Addresss/Pattern
   ```

   Note: The *Address* parameter is followed by the **s** subcommand.
   where the *Address* parameter is the line number, range of line numbers, or symbolic address of the lines you want to delete the pattern from, and the *Pattern* parameter is a character string or regular expression that represents the text you want to delete.
2. To delete the *first instance* of the pattern from each line, type:

   ```
   //
   ```

   OR

   To delete *every instance* of the pattern from each line, type:

   ```
   //g
   ```
3. If you want to display the deletion, type one of the following optional subcommands:

   ```
   l
   ```

   ```
   n
   ```

   ```
   p
   ```
4. Press Enter.

To Delete Text from Lines That Contain a Specified Pattern:

1. Type the following subcommand:

   ```
   [Address]g/Pattern/s
   ```

   where *Address* is an optional parameter that specifies the line number, range of line numbers, or symbolic address of the lines that contains a specified pattern, and the *Pattern* parameter is a character string or regular expression that represents the text you want to find and delete. If you omit the *Address* parameter, the ed editor searches all lines in the file for the pattern.
2. To delete the *first instance* of the pattern from each line that contains it, type:

   ```
   ///
   ```

   OR

   To delete *every instance* of the pattern from each line that contains it, type:

   ```
   ///g
   ```
3. If you want to display the deletion, type one of the following optional subcommands:

   ```
   l
   ```

   ```
   n
   ```

   ```
   p
   ```
4. Press Enter.

To Delete a Pattern from Lines That Contain a Different Specified Pattern:

1. Type the following subcommand:
   ```
   [Address]g/SearchPattern/s
   ```

   where *Address* is an optional parameter that specifies the line number, range of line numbers, or symbolic address of the lines that contains a specified pattern, and the *SearchPattern* parameter is a character string or regular expression that represents text that is in the lines you want to change. If you omit the *Address* parameter, the ed editor searches all lines in the file for the specified pattern.
2. To specify the text you want to delete, type:
   ```
   /DeletePattern/
   ```
3. To delete the *first instance* of the pattern from each line, type:
   ```
   /
   ```

   OR

   To delete *every instance* of the pattern from each line, type:
   ```
   /g
   ```

   Note: The entire subcommand string looks like this:
   ```
   [Address]g/SearchPattern/s/DeletePattern//[g]
   ```
4. If you want to display the deletion, type one of the following optional subcommands:

   ```
   l
   ```

   ```
   n
   ```

   ```
   p
   ```
5. Press Enter.

For example, to delete the first instance of a pattern from lines that contain a different specified pattern, type:

```
1,.g/rem/s/tmp//l
```

The previous example searches from the first line to the current line (`1,.`) for all lines that contain (`g`) the `rem` character string. It deletes the first instance of the `tmp` character string from each of those lines (`/`), then displays the lines that had text deleted from them, including the nonprinting characters in those lines (`l`).

To Delete a Pattern from Lines That Do Not Contain a Different Specified Pattern:

1. Type the following subcommand:
   ```
   [Address]v/SearchPattern/s
   ```

   where *Address* is an optional parameter that specifies the line number, range of line numbers, or symbolic address of the lines that contains a specified pattern, and the *SearchPattern* parameter is a character string or regular expression that represents text that is not in the lines you want to find and change. If you omit the *Address* parameter, the ed editor searches all lines in the file for the specified pattern.
2. To specify the text you want to delete, type:
   ```
   /DeletePattern/
   ```
3. To delete the *first instance* of the pattern, type:
   ```
   /
   ```

   OR

   To delete *every instance* of the pattern from each line, type:

```
/g
```

Note: The entire subcommand string looks like this:

```
[Address]v/SearchPattern/s/DeletePattern//[g]
```

4. If you want to display the deletion, type one of the following optional subcommands:

```
l

n

p
```

5. Press Enter.

For example, to delete the first instance of a pattern from lines that do not contain a specified pattern, type:

```
1,.v/rem/s/tmp//l
```

The previous example searches from the first line to the current line (`1,.`) for all lines that do not contain (`v`) the `rem` character string. It deletes the first instance of the `tmp` character string from each of those lines (`/`), then displays the lines that had text deleted from them, including the nonprinting characters in those lines (`l`).

Displaying Text

(.,.)**l** The **l** (list) subcommand writes the addressed lines to standard output in a visually unambiguous form and writes the characters \\\, \\a, \\b, \\f, \\r, \\t, and \\v in the corresponding escape sequence. The **l**subcommand writes nonprintable characters as one 3-digit octal number, with a preceding \ (backslash) for each byte in the character (most significant byte first).

The **l** subcommand wraps long lines, and you can indicate the wrap point by writing the \ (backslash)/new-line character sequence. Wrapping occurs at the 72nd column position. The $ (dollar sign) marks the end of each line. You can append the **l** subcommand to any ed editor subcommand except the **e**, **E**, **f**, **q**, **Q**, **r**, **w**, or **!** subcommand. The current line number is set to the address of the last line written.

(.,.)**n** The **n** (number) subcommand displays the addressed lines, each preceded by its line number and a tab character (displayed as blank spaces); **n** sets the current line to the last line displayed. You can append the **n** subcommand to any ed editor subcommand except **e**, **f**, **r**, or **w**. For example, the **dn** subcommand deletes the current line and displays the new current line and line number.

(.,.)**p** The **p** (print) subcommand displays the addressed lines and sets the current line to the last line displayed. You can append the **p** subcommand to any ed editor subcommand except **e**, **f**, **r**, or **w**. For example, the **dp** subcommand deletes the current line and displays the new current line.

(.)**=** Without an address, the **=** (equal sign) subcommand displays the current line number. When preceded by the $ address, the **=** subcommand displays the number of the last line in the buffer. The **=** subcommand does not change the current line and cannot be appended to a **g** subcommand or **v** subcommand.

When you search for lines that contain or do not contain a specified pattern, you can select a range of line numbers to search. You can select and display one line or a group of lines in an ed editor file several different ways. Use the preceding format to perform the following editing tasks:

- Displaying an addressed line or group of lines
- Displaying an addressed line or group of lines and their nonprinting characters
- Displaying an addressed line or group of lines and their line numbers
- Displaying lines that contain a search pattern
- Displaying lines that contain a search pattern, including their nonprinting characters
- Displaying lines that contain a search pattern, including their line numbers
- Displaying lines that do not contain a search pattern

- Displaying lines that do not contain a search pattern, including their nonprinting characters
- Displaying lines that do not contain a search pattern, including their line numbers

To Display an Addressed Line or Group of Lines: Type the following subcommand:

```
Addressp
```

where the *Address* parameter is the line number or symbolic address of the lines you want to display.

The line or lines addressed are displayed on the screen. If the group of lines is too long to fit on the screen, the ed editor displays as many as will fit, beginning with the first line addressed.

To Display an Addressed Line or Group of Lines and Their Nonprinting Characters: Type the following subcommand:

```
Addressl
```

where the *Address* parameter is the line number or symbolic address of the lines you want to display.

The line or lines addressed and their nonprinting characters are displayed on the screen. If the group of lines is too long to fit on the screen, the ed editor displays as many as will fit, beginning with the first line addressed.

To Display an Addressed Line or Group of Lines and Their Line Numbers: Type the following subcommand:

```
Addressn
```

where the *Address* parameter is the line number or symbolic address of the lines you want to display.

The line or lines addressed are displayed on the screen. The line number for each line is displayed beside the line. If the group of lines is too long to fit on the screen, the ed editor displays as many as will fit, beginning with the first line addressed.

To Display Lines That Contain a Search Pattern: Type the following subcommand:

```
Addressg/Pattern/p
```

where the *Address* parameter is the range of lines and the *Pattern* parameter is the character string or regular expression that you want to search.

The line or lines that contain the specified pattern are displayed on the screen. If the group of lines is too long to fit on the screen, the ed editor displays as many as will fit, beginning with the first line addressed.

To Display Lines That Contain a Search Pattern, Including Their Nonprinting Characters: Type the following subcommand:

```
[Address]g/Pattern/l
```

where *Address* is an optional parameter that specifies the range of lines and the *Pattern* parameter is the character string or regular expression that you want to search. If you omit the *Address* parameter, the ed editor searches the entire file.

The line or lines that contain the specified pattern are displayed on the screen. Nonprinting characters show up in the display. If the group of lines is too long to fit on the screen, the ed editor displays as many as will fit, beginning with the first line addressed.

To Display Lines That Contain a Search Pattern, Including Their Line Numbers: Type the following subcommand:

```
[Address]g/Pattern/n
```

where *Address* is an optional parameter that specifies the range of lines and the *Pattern* parameter is the character string or regular expression that you want to search. If you omit the *Address* parameter, the ed editor searches the entire file.

The line or lines that contain the specified pattern are displayed on the screen. The line number for each line is displayed beside the line. If the group of lines is too long to fit on the screen, the ed editor displays as many as will fit, beginning with the first line addressed.

To Display Lines That Do Not Contain a Search Pattern: Type the following subcommand:

```
[Address]v/Pattern/p
```

where *Address* is an optional parameter that specifies the range of lines and the *Pattern* parameter is the character string or regular expression that you want to search. If you omit the *Address* parameter, the ed editor searches the entire file.

The line or lines that do not contain the specified pattern are displayed on the screen. If the group of lines is too long to fit on the screen, the ed editor displays as many as will fit, beginning with the first line addressed.

To Display Lines That Do Not Contain a Search Pattern, Including Their Nonprinting Characters: Type the following subcommand:

```
[Address]v/Pattern/l
```

where *Address* is an optional parameter that specifies the range of lines and the *Pattern* parameter is the character string or regular expression that you want to search. If you omit the *Address* parameter, the ed editor searches the entire file.

The line or lines that do not contain the specified pattern are displayed on the screen, including the nonprinting characters. If the group of lines is too long to fit on the screen, the ed editor displays as many as will fit, beginning with the first line addressed.

To Display Lines That Do Not Contain a Search Pattern, Including Their Line Numbers: Type the following subcommand:

```
[Address]v/Pattern/n
```

where *Address* is an optional parameter that specifies the range of lines and the *Pattern* parameter is the character string or regular expression that you want to search. If you omit the *Address* parameter, the ed editor searches the entire file.

The line or lines that do not contain the specified pattern are displayed on the screen, along with their line numbers. If the group of lines is too long to fit on the screen, the ed editor displays as many as will fit, beginning with the first line addressed.

Joining and Splitting Lines

(.,.+1)**j** [**l**] [**n**] [**p**]

The **j** (join) subcommand joins contiguous lines by removing the intervening new-line characters. If given only one address, the **j** subcommand does nothing.

Type the **l** (list), **n** (number), or **p** (print) subcommand if you want to display the joined lines. These subcommands are optional.

The ed editor provides several ways to join or split a line. Use the preceding format to perform the following editing tasks:

- Joining the current and next lines

- Joining addressed lines
- Splitting the current line
- Splitting an addressed line

To Join the Current and Next Lines: Type the following subcommand:

```
j[l][n][p]
```

where **l**, **n**, and **p** are optional subcommands that display the joined lines.

To Join Addressed Lines: Type the following subcommand:

```
Addressj[l][n][p]
```

where the *Address* parameter is a set of contiguous lines that will form one line, and **l**, **n**, and **p** are optional subcommands that display the joined lines.

To Split the Current Line:

1. To split the current line after a specified pattern, type the following subcommand:

   ```
   s/Pattern/Pattern\
   ```

   where the *Pattern* parameter is the character string that you want to split the line after.

   Note: Make sure that both strings represented by the *Pattern* parameter are exactly alike.
2. Press Enter.
3. Type the following backslash:

   ```
   /
   ```
4. To display the split line, type one of the following optional subcommands:

   ```
   l
   ```

   ```
   n
   ```

   ```
   p
   ```
5. Press Enter.

To Split an Addressed Line:

1. To split an addressed line after a specified pattern, type the following subcommand:

   ```
   Addresss/Pattern/Pattern\
   ```

   where the *Address* parameter is the address of the line to split, and the *Pattern* parameter is the character string to split the line after.

   Note: Make sure that both strings represented by the *Pattern* parameter are exactly alike.
2. Press Enter.
3. Type the following backslash:

   ```
   /
   ```
4. To display the split line, type one of the following optional subcommands:

   ```
   l
   ```

   ```
   n
   ```

   ```
   p
   ```

5. Press Enter.

Making Global Changes

(1,$)**g/***Pattern***/***SubcommandList* [**l**] [**n**] [**p**]

The **g** (global) subcommand first marks every line that matches the *Pattern* parameter. The pattern can be a fixed character string or a regular expression. Then, for each marked line, this subcommand sets the current line to the marked line and runs the *SubcommandList* parameter. Enter a single subcommand or the first subcommand of a list of subcommands on the same line with the **g** subcommand; enter subsequent subcommands on separate lines. Except for the last line, each of the lines should end with a \ (backslash).

The *SubcommandList* parameter can include the **a**, **i**, and **c** subcommands and their input. If the last command in the *SubcommandList* parameter would usually be the **.** (period) that ends input mode, the **.** (period) is optional. If no *SubcommandList* parameter exists, the current line is displayed. The *SubcommandList* parameter cannot include the **g** , **G**, **v**, or **V** subcommand.

Type the **l** (list), **n** (number), or **p** (print) subcommand if you want to display the changes. These subcommands are optional.
Note: The **g** subcommand is similar to the **v** subcommand, which runs the *SubcommandList* parameter for every line that does not contain a match for the pattern.

(1,$)**G/***Pattern***/** [**l**] [**n**] [**p**]

The interactive **G** (Global) subcommand marks every line that matches the *Pattern* parameter, displays the first marked line, sets the current line to that line, and then waits for a subcommand. A pattern can be a fixed character string or a regular expression.

The **G** subcommand does not accept the **a**, **i**, **c**, **g**, **G**, **v**, and **V** subcommands. After the subcommand finishes, the **G** subcommand displays the next marked line, and so on. The **G** subcommand takes a new-line character as a null subcommand. A :& (colon ampersand) causes the **G** subcommand to run the previous subcommand again. You can stop the **G** subcommand by pressing Ctrl+C.

Type the **l** (list), **n** (number), or **p** (print) subcommand if you want to display the changes. These subcommands are optional.

(1,$)**v/***Pattern***/***SubcommandList* [**l**] [**n**] [**p**]

The **v** subcommand runs the subcommands in the *SubcommandList* parameter for each line that does not contain a match for the *Pattern* parameter. A pattern can be a fixed character string or a regular expression.

Type the **l** (list), **n** (number), or **p** (print) subcommand if you want to display the changes. These subcommands are optional.

The **v** subcommand does not accept the **a**, **i**, **c**, **g**, **G**, and **V** subcommands.
Note: The **v** subcommand complements the **g** subcommand, which runs the *SubcommandList* parameter for every line that contains a match for the pattern.

(1,$)**V/***Pattern***/** [**l**] [**n**] [**p**]

The **V** subcommand marks every line that does not match the *Pattern* parameter, displays the first marked line, sets the current line to that line, and then waits for a subcommand. A pattern can be a fixed character string or a regular expression.

Type the **l** (list), **n** (number), or **p** (print) subcommand if you want to display the changes. These subcommands are optional.

The **V** subcommand does not accept the **a**, **i**, **c**, **g**, **G**, and **v** subcommands.
Note: The **V** subcommand complements the **G** subcommand, which marks the lines that match the pattern.

Marking Text

(.)**k***x* [**l**] [**n**] [**p**]

The **k** (mark) subcommand marks the addressed line with the name specified by the *x* parameter, which must be a lowercase ASCII letter. The address '*x* (single quotation mark before the marking character) then addresses this line. The **k** subcommand does not change the current line.

Type the **l** (list), **n** (number), or **p** (print) subcommand if you want to display the marked text. These subcommands are optional.

To Mark the Current Line: Type the following subcommand:

```
kLetter[l][n][p]
```

where the *Letter* parameter is the letter *a* through *z* for a mark, and **l**, **n**, and **p** are optional subcommands that display the marked text.

To Mark an Addressed Line: Type the following subcommand:

```
AddresskLetter[l][n][p]
```

where the *Address* parameter is the line number or symbolic address of the line you want to mark, and the *Letter* parameter is the letter *a* through *z* for a mark. The **l**, **n**, and **p** optional subcommands display the marked text.

Moving Text

(.,.)**m***A* [**l**] [**n**] [**p**]

The **m** (move) subcommand repositions the addressed line or lines. The first moved line follows the line addressed by the *A* parameter. A parameter of 0 moves the addressed line or lines to the beginning of the file. The address specified by the *A* parameter cannot be one of the lines to be moved. The **m** subcommand sets the current line to the last moved line.

Type the **l** (list), **n** (number), or **p** (print) subcommands if you want to display the deletion. These subcommands are optional.

Moving a line or a set of lines deletes the specified lines from their original location and places them in a new location. You can select which lines to move by address or pattern. Use the preceding format to perform the following editing tasks:

- Moving the current line
- Moving lines specified by address
- Moving lines that contain a specified pattern
- Moving lines that do not contain a specified pattern

To Move the Current Line: Type the following subcommand:

```
mAddress[l][n][p]
```

where the *Address* parameter is the line number or symbolic address of the line you want the current line to follow, and **l**, **n**, and **p** are optional subcommands that display the moved line.

To Move Lines Specified by Address: Type the following subcommand:

```
LineNumbermDestinationAddress[l][n][p]
```

where the *LineNumber* parameter is the address of the lines you want to move, and the *DestinationAddress* parameter is the line you want the moved lines to follow. The **l**, **n**, and **p** optional subcommands display the moved lines.

To Move Lines That Contain a Specified Pattern: Type the following subcommand:

```
[Address]g/Pattern/m[DestinationAddress][l][n][p]
```

where *Address* is an optional parameter that specifies the range of lines to search for lines that contain the specified pattern, the *Pattern* parameter is the text you are searching for, and *DestinationAddress* is an optional parameter that represents the line you want the moved lines to follow. The **l**, **n**, and **p** optional subcommands display the moved lines.

If you omit the *Address* parameter, the ed editor searches the entire file for lines that contain the pattern. If you omit the *DestinationAddress* parameter, the moved text is placed after the current line.

To Move Lines That Do Not Contain a Specified Pattern: Type the following subcommand:

```
[Address]v/Pattern/m[DestinationAddress][l][n][p]
```

where *Address* is an optional parameter that specifies the range of lines to search for lines that do not contain the specified pattern, the *Pattern* parameter is the text, and *DestinationAddress* is an optional parameter that represents the line you want the moved text to follow. The **l**, **n**, and **p** optional subcommands display the moved lines.

If you omit the *Address* parameter, the ed editor searches the entire file for lines that do not contain the pattern. If you omit the *DestinationAddress* parameter, the moved text is placed after the current line.

Saving Text

(1,$)**w** *File*

The **w** (write) subcommand copies the addressed lines from the buffer to the file specified by the *File* parameter. If the file does not exist, the **w** subcommand creates it with permission code 666 (read and write permission for everyone), unless the **umask** setting specifies another file creation mode.

The **w** subcommand does not change the default file name (unless the *File* parameter is the first file name used since you started the ed editor). If you do not provide a file name, the **w** subcommand uses the default file name. The **w** subcommand does not change the current line.

If the ed editor successfully writes the file from the buffer, it displays the number of characters written. If you specify the **!** *Command* subcommand instead of a file name, the **w** subcommand reads the output of the operating system command specified by the *Command* parameter. The **w** subcommand does not save the name of the operating system command you specified as a default file name.
Note: Because 0 is not a legal address for the **w** subcommand, you cannot create an empty file with the **ed** command.

You can save changes to a file in several ways. Use the preceding format to perform the following actions:

- Saving a file to the current file
- Saving part of a file to the current file
- Saving a file to a different file
- Saving part of a file to a different file

To Save a File to the Current File: Type the following subcommand:

```
w
```

The current file is saved under its current name, and the ed editor displays the number of characters written.

To Save Part of a File to the Current File: Type the following subcommand:

```
Addressw
```

where the *Address* parameter specifies the line or group of lines to write. The ed editor displays the number of characters written.

To Save a File to a Different File: Type the following subcommand:

```
w File
```

where the *File* parameter is the name of the file to write to.

The current file is saved to the file specified by the *File* parameter. The ed editor displays the number of characters written.

To Save Part of a File to a Different File: Type the following subcommand:

```
Addressw File
```

where the *Address* parameter specifies the line or group of lines to write and the *File* parameter specifies the file to write to.

The specified lines are saved to the file specified by the *File* parameter. The ed editor displays the number of characters written.

Searching Text

You can search forward or backward from the current line for a pattern of text. The pattern can be a character string or a regular expression made up of literal characters and the special characters ^ (circumflex), $ (dollar sign), . (period), [(left bracket),] (right bracket), * (asterisk), \ (backslash), % (percent sign), and the & key.

You can use the ed editor to perform the following text searches:

- Searching forward
- Searching backward
- Repeating a search in the same direction
- Repeating a search in the opposite direction

To Search Forward: Type the following subcommand:

```
/Pattern
```

where the *Pattern* parameter is a character string or regular expression that specifies the text to search for.

The cursor moves to the first character of the text specified by the pattern.

To Search Backward: Type the following subcommand:

```
?Pattern
```

where the *Pattern* parameter is a character string or regular expression that specifies the text to search for.

The cursor moves to the first character of the text specified by the pattern.

To Repeat a Search in the Same Direction: Type the following subcommand:

```
/
```

The cursor moves to the first character of the closest instance of the text specified by the pattern in the last search command.

To Repeat a Search in the Opposite Direction: Type the following subcommand:

?

The cursor moves to the first character of the closest instance of the text specified by the pattern in the last search command.

Substituting Text

(.,.)**s/***Pattern***/***Replacement***/** [**l**] [**n**] [**p**]
(.,.)**s/***Pattern***/***Replacement***/ng** [**l**] [**n**] [**p**]

The **s** (substitute) subcommand searches each addressed line for a string that matches the *Pattern* parameter and replaces the string with the specified *Replacement* parameter. A pattern can be a fixed character string or a regular expression. Without the global subcommand (**g**), the **s** subcommand replaces only the first matching string on each addressed line. With the **g** subcommand, the **s** subcommand replaces every occurrence of the matching string on each addressed line. If the **s** subcommand does not find a match for the pattern, it returns the error message ? (question mark).

Type the **l** (list), **n** (number), or **p** (print) subcommand to display the substituted text. These subcommands are optional.
Note: Any character except a space or a new-line character can separate (delimit) the *Pattern* and *Replacement* parameters. The **s** subcommand sets the current line to the last line changed.

If the *Number* parameter (an integer) is specified, then the first number that matches strings in each addressed line is replaced.

An & (ampersand) character used in the *Replacement* parameter has the same value as the *Pattern* parameter. For example, the subcommand **s/are/&n't/** has the same effect as the subcommand **s/are/aren't/** and replaces **are** with **aren't** on the current line. A \& (backslash, ampersand) removes the special meaning of the & character in the *Replacement* parameter.

A subpattern is part of a pattern enclosed by the strings \((backslash, left parenthesis) and \) (backslash, right parenthesis); the pattern works as if the enclosing characters were not present. In the *Replacement* parameter, *Number* refers to strings that match subpatterns. For example, the **s/\(t\)\(h\)\(e\)/t\1\2ose)** subcommand replaces **the** with **those** if a match for the pattern **the** exists on the current line. Whether subpatterns are nested or in a series, *Number* refers to the occurrence specified by the *Number* parameter, counting from the left of the delimiting characters, \) (backslash, right parenthesis).

The % (percent sign), when used alone as the *Replacement* parameter, causes the **s** subcommand to repeat the previous *Replacement* parameter. The % does not have this special meaning if it is part of a longer *Replacement* parameter or if it is preceded by a \ (backslash).

You can split lines by substituting new-line characters into them. In the *Replacement* parameter. Pressing the \+Enter key sequence quotes the new-line character (not displayed) and moves the cursor to the next line for the remainder of the string. New-line characters cannot be substituted as part of a **g** subcommand or **v** subcommand list.

The ed editor provides several ways to substitute text. Use the preceding format to perform the following editing tasks:

- Substituting text within the current line
- Substituting text within an addressed line or group of lines
- Substituting a specified pattern within lines that contain that pattern
- Substituting a pattern within lines that contain a different pattern
- Substituting a pattern within lines that do not contain a different pattern

To Substitute Text within the Current Line:

1. Type the following subcommand:

   ```
   s/OldString/NewString
   ```

   where the *OldString* parameter is the existing text and the *NewString* parameter is the text you want to substitute for it.
2. Type one of the following actions:

   To substitute the *NewString* parameter for the first instance of the *OldString* parameter within the current line, type:

   ```
   /
   ```

   To substitute the *NewString* parameter for every instance of the *OldPattern* parameter within the current line, type:

   ```
   /g
   ```
3. To display the changed text, type one of the following optional subcommands:

   ```
   l
   ```

   ```
   n
   ```

   ```
   p
   ```
4. Press Enter.

To Substitute Text within an Addressed Line or Group of Lines:

1. Type the following subcommand:

   ```
   Addresss/OldPattern/NewString
   ```

   where the *Address* parameter is the address of the line or group of lines where you want to substitute text, the *OldPattern* parameter is the existing text, and the *NewString* parameter is the text you want to substitute.
2. Type one of the following actions:

   To substitute the *NewString* parameter for the first instance of the *OldPattern* parameter within each line, type:

   ```
   /NewString/
   ```

   To substitute the *NewString* parameter for every instance of the *OldPattern* parameter within each line, type:

   ```
   /NewString/g
   ```

   To substitute the *NewString* parameter for the first instance of the *NumberOldPattern* parameter on each address line, type:

   ```
   /NewString/Number
   ```
3. To display the changed text, type one of the following optional subcommands:

   ```
   l
   ```

   ```
   n
   ```

   ```
   p
   ```
4. Press Enter.

To Substitute a Specified Pattern within Lines That Contain That Pattern:

1. Type the following subcommand:

   ```
   Addressg/Pattern/s//NewString
   ```

   where the *Address* parameter is the address of the group of lines that you want to search for the pattern specified with the *Pattern* parameter, and the *NewString* parameter is the text you want to substitute for the *Pattern* parameter.
2. Type one of the following actions:

   To substitute the *NewString* parameter for the first instance of the *Pattern* parameter within each line, type:

   ```
   /
   ```

   To substitute the *NewString* parameter for every instance of the *Pattern* parameter within each line, type:

   ```
   /g
   ```
3. To display the changed text, type one of the following optional subcommands:

   ```
   l

   n

   p
   ```
4. Press Enter.

To Substitute a Pattern within Lines That Contain a Different Pattern:

1. Type the following subcommand:

   ```
   Addressg/Pattern/s/OldString/NewString
   ```

   where the *Address* parameter is the address of the group of lines that you want to search for the pattern specified with the *Pattern* parameter, the *OldString* parameter is the text you want to replace, and the *NewString* parameter is the text you want to substitute in place of the *OldString* parameter.
2. Type one of the following actions:

   To substitute the *NewString* parameter for the first instance of the *OldString* parameter within each line that contains the *Pattern* parameter, type:

   ```
   /
   ```

   To substitute the *NewString* parameter for every instance of the *OldString* parameter within each line that contains the *Pattern* parameter, type:

   ```
   /g
   ```
3. To display the changed text, type one of the following optional subcommands:

   ```
   l

   n

   p
   ```
4. Press Enter.

To Substitute a Pattern within Lines That Do Not Contain a Different Pattern:

1. Type the following subcommand:

   ```
   Addressv/Pattern/s/OldString/NewString
   ```

where the *Address* parameter is the address of the group of lines that you want to search for the pattern specified with the *Pattern* parameter, the *OldString* parameter is the text you want to replace, and the *NewString* parameter is the text you want to substitute in place of the *OldString* parameter.

2. Type one of the following actions:

   To substitute the *NewString* parameter for the first instance of the *OldString* parameter within each line that does not contain the *Pattern* parameter, type:

   ```
   /
   ```

   To substitute the *NewString* parameter for every instance of the *OldString* parameter within each line that does not contain the *Pattern* parameter, type:

   ```
   /g
   ```

3. To display the changed text, type one of the following optional subcommands:

   ```
   l
   ```

   ```
   n
   ```

   ```
   p
   ```

4. Press Enter.

Undoing Text Changes

u [**l**] [**n**] [**p**]

The **u** (undo) subcommand restores the buffer to the state it was in before it was last modified by an ed editor subcommand. The **u** subcommand cannot undo the **e**, **f**, and **w** subcommands.

Type the **l** (list), **n** (number), or **p** (print) subcommand if you want to display the changes. These subcommands are optional.

To Undo Text Changes: Type the following subcommand:

```
u[l][n][p]
```

where **l**, **n**, and **p** are optional subcommands that display the changes. All add, change, move, copy, or delete editing functions performed to the text after the last save are undone.

Manipulating Files

You can use ed editor subcommands to manipulate files to perform the following tasks:

- Adding another file to the current file
- Changing the default file name
- Editing additional files

Adding Another File to the Current File

(**$**)**r** *File*

The **r** (read) subcommand reads a file into the buffer after the addressed line. The **r** subcommand does not delete the previous contents of the buffer. When entered without the *File* parameter, the **r** subcommand reads the default file, if any, into the buffer. The **r** subcommand does not change the default file name.

A 0 address causes the **r** subcommand to read a file in at the beginning of the buffer. After it reads a file successfully, the **r** subcommand displays the number of characters read into the buffer and sets the current line to the last line read.

If ! (exclamation point) replaces the *File* parameter in an **r** subcommand, the rest of the line is taken as an operating system shell command whose output is to be read. The **r** subcommand does not store the names of operating system commands as default file names.

To Insert a File after the Current Line: Type the following subcommand:

```
r File
```

where the *File* parameter is the name of the file to be inserted.

The ed editor reads the file specified by the *File* parameter into the current file after the current line and displays the number of characters read into the current file.

To Insert a File after a Line Specified by Address: Type the following subcommand:

```
Addressr File
```

where the *Address* parameter specifies the line that you want the inserted file to follow, and the *File* parameter is the name of the file to be inserted.

The ed editor reads the file specified by the *File* parameter into the current file after the specified line and displays the number of characters read into the current file.

Changing the Default File Name

f [*File*] The **f** (file name) subcommand changes the default file name (the stored name of the last file used) to the name specified by the *File* parameter. If a *File* parameter is not specified, the **f** subcommand displays the default file name. (The **e** subcommand stores the default file name.)

To Display the Name of a File: Type the following subcommand:

```
f
```

The ed editor displays the name of the file in the edit buffer.

To Name a File: Type the following subcommand:

```
f File
```

where the *File* parameter is the new name for the file in the edit buffer.

The file in the edit buffer is renamed.

Editing Additional Files

e *File* The **e** (edit) subcommand first deletes any contents from the buffer, sets the current line to the last line of the buffer, and displays the number of characters read into the buffer. If the buffer has been changed since its contents were saved (with the **w** subcommand), the ed editor displays a ? (question mark) before it clears the buffer.

The **e** subcommand stores the *File* parameter as the default file name to be used, if necessary, by subsequent **e**, **r**, or **w** subcommands. (To change the name of the default file name, use the **f** subcommand.)

When an ! (exclamation point) replaces the *File* parameter, the **e** subcommand takes the rest of the line as an operating system shell command and reads the command output. The **e** subcommand does not store the name of the shell command as a default file name.

E *File* The **E** (Edit) subcommand works like the **e** subcommand with one exception; the **E** subcommand does not check for changes made to the buffer after the last **w** subcommand. Any changes you made before re-editing the file are lost.

You can use the **e** or **E** subcommands to perform the following tasks:

- Re-editing the current file without saving it
- Re-editing the current file after saving it
- Editing a file after the current file Is saved
- Editing a file without saving the current file

To Re-Edit the Current File without Saving It: Type the following subcommand:

```
E
```

The ed editor displays the number of characters in the file. Any changes you made before re-editing the file are lost.

To Re-Edit the Current File after Saving It: Type the following subcommand:

```
e
```

The ed editor displays the number of characters in the file.

To Edit a File after the Current File Is Saved: Type the following subcommand:

```
e File
```

where the *File* parameter is the name of a new or existing file that you want to edit.

For an existing file, the ed editor displays the number of characters in the file. For a new file, the ed editor displays a ? (question mark) and the name of the file.

To Edit a File without Saving the Current File: Type the following subcommand:

```
E File
```

where the *File* parameter is the name of a new or existing file that you want to edit.

For an existing file, the editor displays the number of characters in the file. For a new file, the ed editor displays a ? (question mark) and the name of the file.

Miscellaneous Functions of the ed Editor Subcommands

You can use ed editor subcommands to perform the following tasks:

- Changing the prompt string
- Entering system commands
- Exiting the ed editor
- Requesting help

Changing the Prompt String

P The **P** (Prompt) subcommand turns on or off the ed editor prompt string, which is represented by an * (asterisk). Initially, the **P** subcommand is turned off.

To Start or Stop Displaying the Prompt String: Type the following subcommand:

```
P
```

The ed editor prompt, an * (asterisk), is displayed or not displayed, depending on its previous setting.

Entering System Commands

! *Command* The **!** subcommand allows you to run operating system commands without leaving the ed editor. Anything that follows the **!** subcommand on an ed editor subcommand line is interpreted as an operating system command. Within the text of that command string, the ed editor replaces the unescaped % (percent sign) with the current file name, if one exists.

You can repeat the previous operating system command by entering an ! (exclamation point) after the **!** ed editor subcommand. If the operating system command interpreter (the **sh** command) expands the command string, the ed editor echoes the expanded line. The **!** subcommand does not change the current line.

You can use the **!** subcommand to perform the following actions:

- Running one operating system command
- Repeating an operating system command
- Running several operating system commands

To Run One Operating System Command: Type the following subcommand:

```
!Command
```

where the *Command* parameter specifies an operating system command usually entered at the prompt.

The command runs and displays its output. After the command completes, the editor displays an **!** (exclamation point).

To Repeat an Operating System Command: Type the following subcommand:

```
!
```

The previously run operating system command runs and displays its output. After the command completes, the editor displays an **!** (exclamation point).

To Run Several Operating System Commands:

1. Type the following subcommand to display an operating system prompt:
   ```
   !sh
   ```
2. Type an operating system command.
3. Press Enter to run the command and display its output.
4. Repeat steps 2 and 3 to run more operating system commands.
5. Press Ctrl+D to return to command mode. The editor displays an **!** (exclamation point).

Exiting the ed Editor

q The **q** (quit) subcommand exits the ed editor after checking whether the buffer has been saved to a file after the last changes were entered. If the buffer has not been saved to a file, the **q** subcommand displays the ? (question mark) message. Enter the **q** subcommand again to exit the ed editor anyway. The changes to the current file are lost.

Q The **Q** (Quit) subcommand exits the ed editor without checking whether any changes were made since the buffer was saved to a file. Any changes made to the buffer since the last save are lost.

To Quit after Checking for Edits:

1. Type the following subcommand:
   ```
   q
   ```
2. If the ed editor displays a ?, type one of the following subcommands:
   To save changes before quitting, type:

```
w
```

then press Enter.

To quit without saving changes, type:

```
q
```

3. Press Enter.

To Quit and Discard Edits:

1. Type the following subcommand:

   ```
   Q
   ```

2. Press Enter. Any changes made to the buffer since the last save are lost.

Requesting Help

h The **h** (help) subcommand provides a brief help message for the most recent ? diagnostic or error message displayed.

H The **H** (Help) subcommand causes the ed editor to display help messages for all subsequent ? diagnostic messages. The **H** subcommand also explains the previous ? if one existed. The **H** subcommand alternately turns this mode on and off; it is initially off.

To Start or Stop Displaying Help Messages: Type the following subcommand:

```
H
```

The help messages are displayed or not displayed for ? responses from the ed editor, depending on the previous setting.

To Display the Last Help Message: Type the following subcommand:

```
h
```

A help message is displayed for the last ? response from the ed editor.

Character Class Support in the ed Editor

In standard *Patterns* expression, a range expression matches the set of all characters that fall between two characters in the collation sequence of the current locale. The syntax of the range expression is as follows:

[*character***-***character***]**

The first character must be lower than or equal to the second character in the collation sequence. For example, [a-c] matches any of the characters a, b, or c in the En_US locale.

The range expression is commonly used to match a character class. For example, [0-9] is used to mean all digits, and [a-z A-Z] is used to mean all letters. This form may produce unexpected results when ranges are interpreted according to the collating sequence in the current locale.

Instead of the preceding form, use a character class expression within [ ] (brackets) to match characters. The system interprets this type of expression according to the character class definition in the current locale. However, you cannot use character class expressions in range expressions.

The syntax of a character class expression is as follows:

[:*CharacterClass***:]**

That is, a left bracket, a colon, the name of the character class, another colon, and then a right bracket.

The following character classes are supported in all locales:

[:upper:]	Uppercase letters
[:lower:]	Lowercase letters
[:alpha:]	Uppercase and lowercase letters
[:digit:]	Digits
[:alnum:]	Alphanumeric characters
[:xdigit:]	Hexadecimal digits
[:punct:]	Punctuation character (neither a control character nor alphanumeric)
[:space:]	Space, tab, carriage return, new-line, vertical tab, or form feed character
[:print:]	Printable characters, including space
[:graph:]	Printable characters, not including space
[:cntrl:]	Control characters
[:blank:]	Space and tab characters

The brackets are part of the character class definition. To match any uppercase ASCII letter or ASCII digit, use the following regular expression:

```
[[:upper:] [:digit:]]
```

Do not use the expression [A-Z0-9].

A locale may support additional character classes.

The newline character is part of the [:space:] character class but will not be matched by this character class. The newline character may only be matched by the special search characters $ (dollar sign) and ^ (caret).

Exit Status

The **ed** and **red** commands return the following exit values:

0	Successful completion.
>0	An error occurred.

Related Information

The **edit** command, **ex** command, **grep** command, **rsh** command, **sed** command, **sh** command, **stty** command, **vi** or **vedit** command, **view** command.

---

edit Command

Purpose

Provides a simple line editor for the new user.

Syntax

edit [**-r**] [*File* ...]

Description

The **edit** command starts a line editor designed for beginning users, a simplified version of the ex editor. The edit editor belongs to a family of editors that includes the ed editor, ex editor, and vi editor. Knowing about the edit editor can help you learn the more advanced features of the other editors. To edit the contents of a file, enter:

```
edit  File
```

When the file specified by the *File* parameter names an existing file, the **edit** command copies it to a buffer and displays the number of lines and characters in it. It then displays a **:** (colon) prompt to show that it is ready to read subcommands from standard input.

If the file specified in the *File* parameter does not already exist, the **edit** command indicates this information and creates the new file. You can specify more than one file name for the *File* parameter, in which case the **edit** command copies the first file into its buffer and stores the remaining file names in an argument list for later use. The edit editor does not make changes to the edited file until you use the **w** subcommand to write the changes.

The edit editor operates in one of the following two modes:

command mode	Recognizes and runs the edit editor subcommands. When you start the edit editor, it is in command mode. To enter command mode at other times, enter only a **.** (period) at the beginning of a line.
text input mode	Allows you to enter text into the edit editor buffer. Enter text input mode by using the **append (a)** subcommand, **change (c)** subcommand, or **insert (i)** subcommand. To end text input mode, enter only a **.** (period) at the beginning of a line.

Flags

-r Recovers the file being edited after an editor or system malfunction.

Addressing Lines in a File

The edit editor uses the following three types of addresses:

- Line number addresses
- Relative position addresses
- Pattern addresses

Line Number Addresses

Line number addresses specify a line within a file by its line number or symbolic name. This method is the simplest way to address a line or lines.

To address the first line by its symbolic name, enter:

```
.
```

To address the last line by its symbolic name, enter:

```
$
```

You also can specify a range of lines by separating the line numbers or symbolic addresses with a comma or a semicolon. The second address must refer to a line that follows the first addressed line in the range.

For example:

```
1,5
```

addresses the lines 1 through 5.

```
.,$
```

addresses the first through the last lines.

Relative Position Addresses

The edit editor can address a line by its relative position to the current line. An address that begins with the -*Number* or +*Number* parameter addresses a line the specified number of lines before or after the current line, respectively.

For example:

```
+8
```

addresses 8 lines after the current line.

You can also address a line relative to the first or last line by using the symbolic names in combination with the -*Number* or +*Number* addresses.

For example:

```
.+3
```

addresses 3 lines after the first line, and:

```
$-10
```

addresses 10 lines before the last line.

Pattern Addresses

You can specify an address line by searching the buffer for a particular pattern. The edit editor searches forward or backward and stops at the first line that contains the match for the *Pattern* parameter. If necessary, the search wraps past the end or beginning of the buffer until it finds a match or returns to the current line.

To search forward, enter:

```
/Pattern/
```

To search backward, enter:

```
?Pattern?
```

You also can specify a range of lines by separating the *Pattern* parameters with a comma or a semicolon. The second address must refer to a line that follows the first addressed line in the range.

For example:

```
Pattern,Pattern
```

The following characters have special meanings when used as part of the *Pattern* parameter:

^ Matches the beginning of a line when used as the first character of the *Pattern* parameter.
$ Matches the end of a line when used as the last character of the *Pattern* parameter.

Using edit Editor Subcommands

The edit editor subcommands affect the current line, which is represented by a **.** (period). When you start the edit editor, the current line is the last line in the buffer. As the buffer is edited, the current line changes to the last line affected by a subcommand. To work with different parts of a file, you must know how to find the current line and how to address different lines in a file.

You can use the edit editor subcommands to perform the following tasks:

- Adding text
- Changing the name of the current file

- Changing text
- Deleting text
- Displaying the current file name and status
- Displaying text and finding the current line
- Editing additional files
- Ending and exiting the edit editor
- Making global changes
- Moving or copying text
- Saving a file after a system crash
- Saving text
- Substituting text
- Undoing a change

Adding Text

In the following subcommands, the *Address* parameter is optional. If you specify an address, do not type the brackets. You can use the full subcommand or its abbreviation, which is shown in parentheses.

[*Address*]**append** (**a**) *Text* **.**
Appends the text you type after the current line if you do not specify an *Address* parameter. You may need to find the current line or specify an address if you are not in the correct position in the buffer.

If you specify an address, the **a** subcommand appends text after the specified line. If you specify a 0 address, the **a** subcommand places the text at the beginning of the buffer.

Type the text, pressing the Enter key at the end of each line. When you have entered all the text, type a **.** (period) alone at the start of a line to end text input mode and return to command mode. You can use the **1,$p** subcommand to display the entire contents of the buffer.

Note: The **a** subcommand differs from the **i** subcommand in the placement of text.

[*Address*]**insert** (**i**)*Text***.**
Inserts text before the current line if you do not specify an *Address* parameter. You may need to find the current line or specify an address if you are not in the correct position in the buffer.

If you specify an address, the **i** subcommand inserts text before the specified line. You cannot specify a 0 address.

Type your text, pressing the Enter key at the end of each line. When you have entered all your text, type a **.** (period) alone at the start of a line to end text input mode and return to command mode. You can use the **1,$p** subcommand to display the entire contents of the buffer.

Note: The **i** subcommand differs from the **a** subcommand in the placement of text.

Changing the Name of the Current File

file *File*
Changes the name of the current file to the name specified by the *File* parameter. The edit editor does not consider this file to be edited.

Changing Text

In the following subcommand, the *Address* parameters are optional. If you specify an address, do not type the brackets. You can use the full subcommand or its abbreviation, which is shown in parentheses.

[*Address1*,*Address2*]**change** (**c**).

Text

Replaces the current line with the text you type if you do not specify the *Address* parameters. You may need to find the current line or specify an address if you are not in the correct position in the buffer.

If you specify an address, the **c** subcommand replaces the addressed line or lines. You can specify a range of lines by separating the addresses with a comma.

Type your text, pressing the Enter key at the end of each line. When you have entered all your text, type a **.** (period) alone at the start of a line to end text input mode and return to command mode. You can use the **1,$p** subcommand to display the entire contents of the buffer. The last input line becomes the current line.

Deleting Text

In the following subcommand, the *Address* and *Buffer* parameters are optional. If you specify an address or buffer, do not type the brackets. You can use the full subcommand or its abbreviation, which is shown in parentheses.

[*Address1*,*Address2*] **delete** [*Buffer*] (**d**)

Deletes the current line if you do not specify the *Address* parameters. You may need to find the current line or specify an address if you are not in the correct position in the buffer.

If you specify an address, the **d** subcommand deletes the addressed line or lines. You can specify a range of lines by separating the addresses with a comma. The line following the last deleted line becomes the current line.

If you specify a buffer by giving a lowercase letter from a to z, the edit editor saves the addressed lines in that buffer. If you specify an uppercase letter, the ed editor appends the lines to that buffer. You can use the **pu** subcommand to put the deleted lines back into the buffer.

Displaying the Current File Name and Status

In the following subcommand, you can use the full subcommand or its abbreviation, which is shown in parentheses.

file (**f**)

Displays the current file name along with the following related information:

- Whether the file was modified since the last **w** subcommand
- Current line number
- Number of lines in the buffer
- Percentage of the buffer indicating the current line location

Displaying Text and Finding the Current Line

In the following subcommands, the *Address* parameters are optional. If you specify an address, do not type the brackets. You can use either the full subcommand or its abbreviation, which is shown in parentheses.

[*Address1*,*Address2*]**number** (**nu**)

Displays the addressed line or lines preceded by its buffer line number. If you do not specify the *Address* parameters, the **nu** subcommand displays the current line and number.

If you specify an address, the **nu** subcommand displays the addressed line or lines. You can specify a range of lines by separating the addresses with a comma. The last line displayed becomes the current line.

[*Address1*,*Address2*]**print** (**p**) Displays the addressed line or lines. If you do not specify the *Address* parameters, the **p** subcommand displays the current line.

If you specify an address, the **p** subcommand displays the addressed line or lines. You can specify a range of lines by separating the addresses with a comma. The last line displayed becomes the current line.

[*Address*]**=** Displays the line number of the addressed line. If you do not specify an *Address* parameter, the **=** subcommand displays the line number of the current line.

[*Address*]**z** Displays a screen of text beginning with the addressed line. If an *Address* parameter is not specified, the **z** subcommand displays a screen of text beginning with the current line.

[*Address*]**z-** Displays a screen of text with the addressed line at the bottom. If an *Address* parameter is not specified, the **z-** subcommand displays a screen of text with the current line at the bottom.

[*Address*]**z.** Displays a screen of text with the addressed line in the middle. If an *Address* parameter is not specified, the **z.** subcommand displays a screen of text with the current line in the middle.

Editing Additional Files

In the following subcommand, you can use the full subcommand or its abbreviation, which is shown in parentheses.

edit *File* (**e**) Begins an editing session on a new file specified by the *File* parameter. The editor first checks to see if the buffer was edited since the last **write (w)** subcommand.

If the file was edited since the last **w** subcommand, the edit editor issues a warning and cancels the **e** subcommand. Otherwise, the edit editor deletes the contents of the editor buffer, makes the named file the current file, and displays the new file name.

After insuring that this file can be edited, the edit editor reads the file into its buffer. If the edit editor reads the file without error, it displays the number of lines and characters that it read. The last line read becomes the new current line.

next (**n**) Copies the next file named in the command line argument list to the buffer for editing.

Ending and Exiting the edit Editor

In the following subcommands, you can use the full subcommand or its abbreviation, which is shown in parentheses.

quit (**q**) Ends the editing session after using the **write (w)** subcommand. If you have modified the buffer and have not written the changes, the edit editor displays a warning message and does not end the editing session.

quit! (**q!**) Ends the editing session, discarding any changes made to the buffer since the last **w** subcommand.

Making Global Changes

In the following subcommand, the *Address* parameters are optional. If you specify an address, do not type the brackets. You can use the full subcommand or its abbreviation, which is shown in parentheses.

[*Address1*,*Address2*]**global/***Pattern*/*SubcommandList* (**g**)

Marks each of the addressed lines that match the *Pattern* parameter. The edit editor then performs the list of subcommands specified in the *SubcommandList* parameter on each marked line.

If you do not specify the *Address* parameters, the **g** subcommand works on the current line. You may need to find the current line or specify an address if you are not in the correct position in the buffer.

If you specify an address, the **g** subcommand works on the addressed line or lines. You can specify a range of lines by separating the addresses with a comma.

A single subcommand or the first subcommand in a subcommand list appears on same line as the **g** subcommand. The remaining subcommands must appear on separate lines, where each line (except the last) ends with a **** (backslash). The default subcommand is the **print (p)** subcommand.

The subcommand list can include the **append (a)** subcommand, **insert (i)** subcommand, and **change (c)** subcommand, and their associated input. In this case, if the ending period is on the last line of the command list, you can omit it.

Note: The **undo (u)** subcommand and the **g** subcommand cannot appear in the subcommand list.

Moving or Copying Text

In the following subcommands, the *Address1* and *Address2* parameters are optional. If you specify an address, do not type the brackets. You must specify the *Address3* parameter. You can use either the full subcommand or its abbreviation, which is shown in parentheses.

[*Address1,Address2*]**move** *Address3* (**m**)
: Moves the current line after the line specified by the *Address3* parameter if you do not specify an address or an address range. You may need to find the current line or specify an address if you are not in the correct position in the buffer.

  If you specify an address, the **m** subcommand moves the addressed line or lines. You can specify a range of addresses by separating the addresses with a comma. The first of the moved lines becomes the current line.

[*Address1,Address2*]**yank** [*Buffer*] (**ya**)
: Copies the specified line or lines into the *Buffer*, an optional parameter specified by a single alpha character a to z. You can use the **pu** subcommand to put these lines into another file.

[*Address*]**put** [*Buffer*] (**pu**)
: Retrieves the contents of the specified *Buffer* parameter and places it after the current line if you do not specify an address. You may need to find the current line or specify an address if you are not in the correct position in the buffer.

  If you specify an address, the **pu** subcommand retrieves the contents of the specified buffer and places it after the addressed line. If you do not specify a *Buffer* parameter, the **pu** subcommand restores the last deleted or copied text.

  You can use the **pu** subcommand with the **delete (d)** subcommand to move lines within a file or with the **yank (ya)** subcommand to duplicate lines between files.

  You cannot use the **pu** and **ya** subcommands inside a macro.

Saving a File after a System Malfunction

preserve
: Saves the current editor buffer as though the system had just malfunctioned. Use this subcommand when a **write (w)** subcommand has resulted in an error and you do not know how to save your work. Use the **recover** subcommand to recover the file.

recover *File*
: Recovers the file specified by the *File* parameter from the system save area. Use this subcommand after a system crash or after a **preserve** subcommand.

Saving Text

In the following subcommand, the *Address* parameters are optional. If you specify an address, do not type the brackets. You can use the full subcommand or its abbreviation, which is shown in parentheses.

[*Address1*,*Address2*]**write** [*File*] (**w**)
: Writes the entire contents of the buffer to the file specified by the *File* parameter if you do not specify an address.

  If you specify an address, the **w** subcommand writes the addressed line or lines to the file specified. You can specify a range of lines by separating the addresses with a comma. The edit editor displays the number of lines and characters that it writes.

  If you do not specify a file, the edit editor uses the current file name. If a *File* parameter does not exist, the editor creates one.

Substituting Text

In the following subcommand, the *Address* parameters are optional. If you specify an address, do not type the brackets. You can use either the full subcommand or its abbreviation, which is shown in parentheses.

[*Address1*,*Address2*] **substitute/***Pattern*/*Replacement*/ (**s**)
[*Address1*,*Address2*] **substitute/***Pattern*/*Replacement*/**g**
: Replaces the first instance of the specified *Pattern* parameter on each addressed line. You can replace every instance of the *Pattern* parameter by adding the **global (g)** subcommand to the end of the **s** subcommand.

  If you do not specify an address, the **s** subcommand works on the current line. You may need to find the current line or specify an address if you are not in the correct position in the buffer. If you specify an address, the **s** subcommand works on the addressed line or lines. You can specify a range of lines by separating the addresses with a comma.

Undoing a Change

In the following subcommand, you can use the full subcommand or its abbreviation, which is shown in parentheses.

undo (**u**)
: Reverses the changes made in the buffer by the last buffer editing subcommand. You cannot undo a **write (w)** subcommand or an **edit (e)** subcommand.

  Note: The **global** subcommands are considered a single subcommand to a **u** subcommand.

Related Information

The **ed** or **red** command, **ex** command, **vi** or **vedit** command.

---

edquota Command

Purpose

Edits user and group quotas.

Syntax

To Edit User Quotas

edquota [**-u**] [**-p** *Proto-UserName*] *UserName* ...

To Edit Group Quotas
edquota [**-g** [**-p** *Proto-GroupName*] *GroupName* ...]

To Edit Change User or Group Grace Period
edquota -t [**-u** | **-g**]

Description
The **edquota** command creates and edits quotas for JFS file systems. For information about how to manage quotas on a JFS2 file system, see the **j2edlimit** command in *AIX Version 6.1 Commands Reference, Volume 3*.

The **edquota** command creates a temporary file that contains current disk quotas of each user and group. It determines the list of file systems with established quotas from the **/etc/filesystems** file. The **edquota** command also invokes the vi editor (or the editor specified by the **EDITOR** environment variable) on the temporary file so that quotas can be added and modified.

Note: If you specify an editor in the **EDITOR** environment variable, you must specify the full path name of the editor.

Quotas are maintained separately for each file system. When you create or edit a quota for a user or a group, the quota applies to a specific file system. A quota must be set in each file system where you want to use quotas.

By default, or when used with the **-u** flag, the **edquota** command edits the quotas of one or more users specified by the *UserName* parameter on the command line. When used with the **-g** flag, the **edquota** command edits the quotas of one or more groups specified by the *GroupName* parameter. The **-p** flag identifies a prototypical user (*UserName*) or a prototypical group *(Proto-GroupName*) and duplicates these quotas for a specified user or group.

A user can exceed established soft limits for a default grace period of 1 week. Upon expiration of the grace period, the soft limit is enforced as a hard limit. The grace period can be specified in days, hours, minutes, or seconds. A value of 0 indicates that the default grace period is imposed; a value of 1 second indicates that no grace period is granted. The **-t** flag changes the grace period.

Fields displayed in the temporary file are:

Blocks in use	The current number of 1KB file system blocks used by this user or group.
Inodes in use	The current number of files used by this user or group.
Block soft limit	The number of 1KB blocks the user or group will be allowed to use during normal operations.
Block hard limit	The total amount of 1KB blocks the user or group will be allowed to use, including temporary storage during a quota grace period.
Inode soft limit	The number of files the user or group will be allowed to create during normal operations.
Inode hard limit	The total number of files the user or group will be allowed to create, including temporary files created during a quota grace period.

Note: A hard limit with a value of 1 indicates that no allocations are permitted. A soft limit with a value of 1, in conjunction with a hard limit with a value of 0, indicates that allocations are permitted only on a temporary basis.

When the editor is exited, the **edquota** command reads the temporary file and modifies the binary quota files to reflect any changes.

Hard or soft limits can only be specified in whole 1 KB block amounts.

Flags

-g Edits the quotas of one or more specified groups.

-p When invoked with the **-u** flag, duplicates the quotas established for a prototypical user for each specified user. When invoked with the **-g** flag, the **-p** flag duplicates the quotas established for a prototypical group for each listed group.

-t Changes the grace period during which quotas can be exceeded before a soft limit is imposed as a hard limit. The default value of the grace period is 1 week. When invoked with the **-u** flag, the grace period is set for all file systems with user quotas specified in the **/etc/filesystems** file. When invoked with the **-g** flag, the grace period is set for all file systems with group quotas specified in the **/etc/filesystems** file.

Note: After changing a grace period using the **edquota** command, the new grace period value will not go into effect until the **quota.user** and **quota.group** files are refreshed by running the **quotaoff** command followed by the **quotaon** command. Users who have already reached their old grace period must reduce their file system usage to a level below their soft limits in order to use the new grace period. In the future, when these users exceed their soft limits, the new grace period will be in effect.

-u Edits the quotas of one or more users.

Note: If the user or group names contains all numbers then it will be treated as a user or group ID. Quotas will then be edited for the ID rather than the name.

Security

Access Control: Only the root user can execute this command.

Attention RBAC users and Trusted AIX users: This command can perform privileged operations. Only privileged users can run privileged operations. For more information about authorizations and privileges, see Privileged Command Database in *Security*. For a list of privileges and the authorizations associated with this command, see the **/etc/security/privcmds** database file.

Examples

To create quotas for user `sharl`, using the quotas established for user `davec` as a prototype, enter:

```
edquota -u -p davec sharl
```

Files

quota.user	Specifies user quotas.
quota.group	Specifies group quotas.
/etc/filesystems	Contains file system names and locations.

Related Information

The **quota** command, **quotacheck** command **quotaon** and **quotaoff** command, **repquota** command.

The Disk Quota System Overview introduces the disk quota system and Setting Up the Disk Quota System describes how to establish disk quotas. Both are in*Security*.

---

efsenable Command

Purpose

Activates Encrypted File System (EFS) capability on a system.

Syntax

efsenable -a [**-v**] [**-k** <*algo*>] [**-f** <*cipher*>] [**-m** <*mode*>] [**-u** <yes|no>] [**-e** <*algo*>]

efsenable -q

Description

The **efsenable** command activates the EFS capability on a system. It creates the EFS administration keystore, the user keystore and the security group keystore. Keystore is a key repository that contains EFS security information. The access key to the EFS administration keystore is stored in the newly created active user's keystore and in the security group keystore. The **efsenable** command creates the **/var/efs** directory. The **/etc/security/user** and **/etc/security/group** files are updated with new EFS attributes. The **efsenable** command also updates the **Config_Rules** ODM database.

Note: The Crypto Library (CLiC) package **clic.rte** must be installed on the system for this command to succeed. This EFS command also requires that Role Based Access Control (RBAC) is enabled on the system, which is the default setting.

Flags

-a Activates the EFS capability on a system.

-v Verbose mode.

-k *algo* Default algorithm for keys. The *algo* flag can be one of the following values:

- RSA_1024 (by default)
- RSA_2048
- RSA_4096

-f *algo* Default cipher for files. The *algo* flag can be one of the following values:

- AES_128_CBC (by default)
- AES_192_CBC
- AES_256_CBC
- AES_128_ECB
- AES_192_ECB
- AES_256_ECB

-m *mode* Default mode for keystores. The *mode* flag can be one of the following values:

- admin (by default)
- guard

-u [yes|no] Specifies if the user can change the mode. Default value is "yes".

-e *algo* Algorithm for the EFS administration key. The possible *algo* values are the same as those of the **-k** flag.

-q Displays the list of available algorithms.

Exit status

0 The command executed successfully.

1 An error occurred during the execution of the command.

2 A syntax error occurred on the command line.

Security

Access Control: Only the root user or a user with the **aix.security.efs** authorization and being a member of the security group can run this command.

Examples

1. To display the available algorithms, enter:
   ```
   efsenable -q
   ```
2. To activate an EFS with default parameters, enter:
   ```
   efsenable –a
   ```
3. To activate an EFS with a non-default algorithm for keys, and cipher for files, enter:
   ```
   efsenable –a –k RSA_4096 –f AES_256_CBC –e RSA_4096
   ```

Files

/etc/security/user	Contains the updates of EFS attributes.
/etc/security/group	Contains the updates of EFS attributes.
/var/efs/users/	Contains the directory for user keystores.
/var/efs/groups/	Contains the directory for group keystores.
/var/efs/efs_admin/	Contains the directory for EFS administration keystore.
/var/efs/efsenabled	Instructs that the EFS is enabled on the system.

Related Information

The **efskeymgr** and **efsmgr** commands.

Securing the base operating system in *Security*.

---

efskeymgr Command

Purpose

Manages user and group repositories for the Encrypted File System keys (or keystores).

Syntax

efskeymgr -?

efskeymgr -q

efskeymgr -V

efskeymgr -C <*group*>

efskeymgr [**-d**] [**-k** <*ks*>] [**-g**] [**-p** <*pw*>] **-v**

efskeymgr [**-d**] [**-k** <*ks*>] **-m**

efskeymgr [**-d**] [**-k** <*ks*>] [**-g**] [**-p** <*pw*>] **-o** <*cmd*>

efskeymgr [**-d**] [**-c** <*cmd*>]

efskeymgr [**-d**] [**-k** <*ks*>] [**-g**] [**-p** <*pw*>] **-n**

efskeymgr [**-d**] [**-k** <*ks*>] [**-g**] [**-p** <*pw*>] **-r** <*mode*>

efskeymgr [**-d**] [**-k** <*ks*>] [**-g**] [**-p** <*pw*>] **-s** <*ks2*>

efskeymgr [**-d**] [**-k** <*ks*>] [**-g**] [**-p** <*pw*>] **-S** <*ks2*>

efskeymgr [**-d**] [**-k** <*ks*>] [**-g**] [**-p** <*pw*>] **-R** <*algo*>

efskeymgr [**-d**] [**-k** <*ks*>] [**-g**] [**-p** <*pw*>] **-D** <*fp*>

efskeymgr [**-d**] [**-k** <*ks*>] [**-g**] [**-p** <*pw*>] **-e** <*file*>

Description

The **efskeymgr** command is dedicated to all key management operations needed by an Encrypted File System (EFS). Once an EFS has been enabled on the system with the **efsenable** command, the keystores (public and private key repositories) are created in the **/var/efs** directory.

The initial password of a user keystore is the user login password. Group keystores and admin keystores are not protected by a password but by an access key. Access keys are stored inside all user keystores that belong to this group.

When you open a keystore (at login or explicitly with the **efskeymgr** command), the private keys contained in this keystore are pushed to the kernel and associated with the process. If access keys are found in the keystore, the corresponding keystores are also opened and the keys are automatically pushed into their kernel.

Keystores support two administration modes: admin mode and guard mode.

admin mode
: When a keystore is set to this mode, an EFS administrator with the **aix.security.efs** RBAC authorization and the access key to admin keystore can open the keystore for management including password reset, key regeneration, access key addition or removal and so on.

guard mode
: When a keystore is set to this mode, the EFS administrator cannot get access to the keystore. In this mode, if the password to keystore is lost, there is no possible recovery of the private key.

When the keystore password is the same as the login password, the keystore is automatically opened at the login time and the keys are available in the session. The keystore password is kept in sync with the login password when the **passwd** command is used and the old password is provided. If at some point the keystore password is not in sync with the login password, you can change the keystore password using the **efskeymgr** command. When the passwords are not synchronized, the keys are no longer automatically associated with the session when you log in.

The following command grants or removes the EFS credentials only for the execution of the *cmd* command. When the *cmd* command returns, the previous process credentials are restored.

```
efskeymgr –o <cmd> and efskeymgr –c <cmd>
```

When a private key is regenerated in a keystore, a new private key is created and the old key is marked "deprecated".

Note: The new key is not pushed into the kernel. You must open your keystore again, either with the **efskeymgr** command or by closing and opening your session, for the new key to be available for file operations.

The deprecated key can still be used to decrypt files, but is no longer used to encrypt files. The deprecated key can be removed from the keystore, but in this case all files that were encrypted with the old key will no longer be accessible.

Note: This EFS command requires that Role Based Access Control (RBAC) is enabled on the system, which is the default setting.

Delayed operations

In some cases, the keystore cannot be modified directly by a command or an action. When this occurs, a special file is created in the keystore directory, and will be parsed next time the keystore is opened. This special file is called a cookie. For keystores in admin mode, the cookies are parsed automatically when the keystore is opened (at login or when the **efskeymgr** command is run). For keystores in guard mode, the cookies are never automatically parsed. The user has to give its approval for each modification of its keystore. When you open a session, a message is displayed if one or more operations are pending on your EFS keystore:

- Your private key must be regenerated.
- You are granted access to **group**/*group1* keystore.

You must run the **efskeymgr -v** command to process pending operations.

The following actions are possible:

- Private key regeneration. This results in a new private key being generated, and the old one being marked ″deprecated″.
- New access key. When you accept this cookie, you obtain access to a new keystore (for example, keystore of a group to which you have been added).
- Remove access key. When you accept this cookie (for example, when the access key is removed from a group), you loose your access to a keystore.

Note: When you run the **efskeymgr** command with any flag that opens your keystore, for example, the **-v** flag, you are prompted what you want to do with each cookie. The choices are as follows:

- Accept the cookie: your keystore is modified according to the cookie, then the cookie is destroyed.
- Postpone the cookie: your keystore is not modified and the cookie is not removed. You will be prompted next time for action.
- Delete the cookie: your keystore is not modified and the cookie is removed. You will have to use the **efskeymgr** command to do the action again.

Flags

General flags:

-d Verbose mode.

-g Does not process pending operations when opening the keystore.

-k *ks* The operation is targeted to the *ks* keystore instead of the active user's keystore. The *ks* value can be as follows:

user/<login>
User <login> keystore.

group/<grpname>
Group <grpname> keystore.

admin/ EFS administration keystore.

-p *pw* Password to use to open the keystore. It is not advised to use this as it can be seen by other users using the **ps** command, for example.

Flags for commands (no access to the keystore files):

-? Displays the command help and exits.

-q Displays a list of supported algorithms for the key regeneration.

-V Displays the keys associated with the active process credentials in the kernel.

Flags for commands (read-only access to keystores):

-c <*cmd*> Removes all keys from the kernel, then runs the *cmd* command. The keys are restored when the *cmd* command terminates.

-m Lists all pending operations on the keystore.

-o <*cmd*> Opens the keystore and pushes the keys, then runs the *cmd* command. The keys are discarded when the *cmd* command terminates.

-v Displays the content of the keystore file.

Flags for commands (read/write access to keystores):

-C <*group*> Creates the keystore of the *group* group.

-D <*fp*> Removes a deprecated private key from the keystore. The *fp* value is the key fingerprint.

-e <*file*> Exports a keystore to a file. The file is PKCS#12 encoded and contains the public and private keys from the keystore. This file can be used in openssh, for example.

-n For user keystores, prompts for a new password for the keystore. For group keystores, generates a new access key and sends to group members. For admin keystores, generates a new access key. The key must then be sent to the EFS administrators with the **efskeymgr** command.

-R <*algo*> Regenerates the keystore private key. See the **-q** flag for the valid values for the *algo* parameter.

-r <*mode*> Changes the keystore administration mode. The *mode* value can be as follows:

admin The EFS administrator can administrate the keystore. Pending operations are applied automatically.

guard The EFS administrator cannot manage the keystore. The user is prompted for any pending operation.

-S <*ks2*> Removes the *ks2* access key from the keystore. On subsequent opening of keystore, the *ks2* private key will no longer be pushed automatically.

-s <*ks2*> Sends the keystore access key to the *ks2* keystore. On subsequent opening of the *ks2* key, the keystore private key will be loaded automatically.

Exit status

0 The command executed successfully.

1 An error occurred during the execution of the command.

2 A syntax error occurred on the command line.

Security

Attention RBAC users and Trusted AIX users: This command can perform privileged operations. Only privileged users can run privileged operations. For more information about authorizations and privileges, see Privileged Command Database in *Security*. For a list of privileges and the authorizations associated with this command, see the **/etc/security/privcmds** database file.

Examples

1. To view your keystore content, enter:
   ```
   efskeymgr –v
   ```
2. To view a user's keystore content, enter:
   ```
   efskeymgr –o ksh
   ```
3. To view the keys associated with the active shell, enter:
   ```
   efskeymgr -V
   ```
4. To regenerate the private key from your keystore, enter:
   ```
   efskeymgr –R RSA_1024
   ```
5. To delete a deprecated key, enter:
   ```
   efskeymgr –D dbb62547:d6925088:45357fd3:54cddbba:27b255a9
   ```
6. To send the access key of the group "students" to the user "joe", enter:
   ```
   efskeymgr –k group/students –s user/joe
   ```

Files

/var/efs	Contains all keystores.
/etc/security/user	Contains the EFS attributes for the creation and management of users keystore.
/etc/security/group	Contains the EFS attributes for the creation of groups keystore.

Related Information

The **efsenable** and **efsmgr** commands.

Securing the base operating system in *Security*.

---

efsmgr Command

Purpose

Manages the files encryption and decryption for the Encrypted File System (EFS).

Syntax

efsmgr -?

efsmgr -q [**-v**]

efsmgr -C <*cipher*> [**-v**]

efsmgr [**-c** <*file*>] **-e** <*file*> [**-v**]

efsmgr [**-c** <*cipher*>] [**-s**] **-E** <*dir*> [**-v**]

efsmgr [**-c** <*cipher*>] **-t** <*file*> [**-v**]

efsmgr [**-c** <*cipher*>] [**-s**] **-T** <*dir*> [**-v**]

efsmgr -d <*file*> [**-v**]

efsmgr [**-s**] **-D** <*dir*> [**-v**]

efsmgr -l <*file*> [**-v**]

efsmgr [**-s**] **-L** <dir> [**-v**]

efsmgr -a <*file*> [**-u** <*user*> | **-g** <*group*>] [**-v**]

efsmgr -r <*file*> [**-u** <*user*> | **-g** <*group*>] [**-v**]

Description

The **efsmgr** command is dedicated to the files encryption management inside EFS. Encrypted files can only be created on the EFS-enabled JFS2 file systems. For more information about enabling EFS on your system, see the **mkfs**, **chfs**, **crfs**, and **efsenable** commands.

There are two ways to create encrypted files: either explicitly by using the following command, or implicitly when inheritance is set on the file system or the directory where the file is being created.

```
efsmgr -e <file>
```

When inheritance is set on a directory, all new files created in this directory are encrypted by default. The cipher used to encrypt files is the inherited cipher. New directories also inherit the same cipher. If inheritance is disabled on a subdirectory, the new files created in this subdirectory will not be encrypted.

When inheritance is set on a file system, all new files created in this file system are encrypted using the inherited cipher. If inheritance is set both on a directory and a file system with different ciphers, new files created in this directory will be encrypted using the cipher inherited from the directory.

Setting or removing inheritance on a directory or a file system has no effect on the existing files. The **efsmgr** command must be used explicitly to encrypt or decrypt files.

The file owner's private key must be loaded into the process before the encrypted file can be created. The access to the encrypted file can be granted to any user or group with a keystore, which is a key repository that contains EFS security information. For more information about managing user and group repositories, see the **efskeymgr** command.

When an encrypted file is being opened, the Discretionary Access Control (DAC) and the Access Control List (ACL) are checked for the file access permission. If the access is granted, the keys loaded into the kernel for the process are searched for a private key matching one of the file's protection keys. If a matching key is found, the file content can be read, otherwise the access is denied.

Note: This EFS command requires that Role Based Access Control (RBAC) is enabled on the system, which is the default setting.

Flags

-c <*cipher*> Uses this cipher instead of the inherited or the default cipher. See the **-q** command for the valid *cipher* values.

-g <*group*> This group must be added or removed from the EFS access list. The *group* value can be either the gid or the group name.

-s The operation is targeted to a file system rather than a directory. In this case, the *dir* parameter must be the mount point of a file system with EFS support.

-u <*user*> This user must be added or removed from the EFS access list. The *user* value can be either the uid or the login name.

-v Verbose mode.

-? Displays the command help and exits.

-a <*file*> Adds access to the specified file to a list of users and groups specified with the **-u** and **-g** flags.

-C <*cipher*> Changes the default cipher for your user to the *cipher* value.

-D <*dir*> Removes the inheritance on the directory. To apply the command on the whole file system, you must add the **-s** flag.

-d <file> Decrypts the specified file.

-E <*dir*> Sets the inheritance on the *dir* directory. To apply the command on the whole file system, you must add the **-s** flag.

-e <file> Encrypts the specified file.

-L <*dir*> Displays the inherited cipher on the specified directory.

-l <*file*> Lists the encryption information of the specified file: cipher, and keys that can decrypt the file.

-q Displays a list of supported ciphers.

-r <*file*> Revokes access to the specified file to a list of users and groups specified with the **-u** and **-g** flags.

-T <*dir*> Changes the inherited cipher on the specified directory. To apply the command on the complete file system, you must add the **-s** flag.

-t <*file*> Refreshes the encryption keys of the specified file. This can also be used to change the file cipher.

Exit status

0 The command executed successfully.

1	An error occurred during the execution of the command.
2	A syntax error occurred on the command line.

Examples

1. To encrypt the **database.txt** file using a strong cipher, enter:
   ```
   efsmgr –e database.txt –c AES_256_CBC
   ```
2. To display the list of keys that can open the file, enter:
   ```
   efsmgr –l database.txt
   ```
3. To add access to user joe and to group maintainers to the file, enter:
   ```
   efsmgr –a database.txt –u joe –g maintainers
   ```
4. To set the inheritance on the file system of the home directory, enter:
   ```
   efsmgr –c AES_128_CBC –s –E /home
   ```

Files

/etc/security/user	Contains the default cipher attributes for the user.

Related Information

The **efsenable** and **efskeymgr** commands.

The **mkfs**, **chfs**, and **crfs** commands.

Securing the base operating system in *Security*.

---

egrep Command

Purpose

Searches a file for a pattern.

Syntax

egrep [**-h**] [**-i**] [**-p**[*Separator*]] [**-s**] [**-v**] [**-w**] [**-x**] [**-y**] [[**-b**] [**-n**] | [**-c** | **-l** | **-q**]] { { **-e***Pattern* | **-f***StringFile* } ... | *Pattern* } [*File ...*]

Description

The **egrep** command searches an input file (standard input by default) for lines matching a pattern specified by the *Pattern* parameter. These patterns are full regular expressions as in the **ed** command (except for the \ (backslash) and \\ (double backslash)). The following rules also apply to the **egrep** command:

- A regular expression followed by a + (plus sign) matches one or more occurrences of the regular expression.
- A regular expression followed by a ? (question mark) matches zero or one occurrence of the regular expression.
- Multiple regular expressions separated by a | (vertical bar) or by a new-line character match strings that are matched by any of the regular expressions.
- A regular expression may be enclosed in ( ) (parentheses) for grouping.

The new-line character will not be matched by the regular expressions.

The order of precedence for operators is [,], *, ?, +, concatenation, | and the new-line character.

> **Note:** The **egrep** command is the same as the **grep** command with the **-E** flag, except that error and usage messages are different and the **-s** flag functions differently.

The **egrep** command displays the file containing the matched line if you specify more than one *File* parameter. Characters with special meaning to the shell ($, *, [, |, ^, (,), \ ) must be in quotation marks when they appear in the *Pattern* parameter. When the *Pattern* parameter is not a simple string, you usually must enclose the entire pattern in single quotation marks. In an expression such as `[a-z]`, the minus means through according to the current collating sequence. A collating sequence may define equivalence classes for use in character ranges. It uses a fast, deterministic algorithm that sometimes needs exponential space.

Notes:

1. Lines are limited to 2048 bytes.
2. Paragraphs (under the **-p** flag) are currently limited to a length of 5000 characters.
3. Do not run the **grep** command on a special file because it produces unpredictable results.
4. Input lines should not contain the NULL character.
5. Input files should end with the newline character.
6. Although some flags can be specified simultaneously, some flags override others. For example, if you specify **-l** and **-n** together, only file names are written to standard output.

Flags

-b	Precedes each line by the block number on which it was found. Use this flag to help find disk block numbers by context. The **-b** flag cannot be used with input from stdin or pipes.
-c	Displays only a count of matching lines.
-e *Pattern*	Specifies a *Pattern*. This works like a simple *Pattern* but is useful when the *Pattern* begins with a - (minus sign).
-f *StringFile*	Specifies a file that contains strings.
-h	Suppresses file names when multiple files are being processed.
-i	Ignores the case of letters when making comparisons.
-l	Lists just the names of files (once) with matching lines. Each file name is separated by a new-line character. If standard input is searched, a path name of `"(StandardInput)"` is returned.
-n	Precedes each line with its relative line number in the file.
-p[*Separator*]	Displays the entire paragraph containing matched lines. Paragraphs are delimited by paragraph separators, as specified by the *Separator* parameter, which are patterns in the same form as the search pattern. Lines containing the paragraph separators are used only as separators; they are never included in the output. The default paragraph separator is a blank line.
-q	Suppresses all output to standard output, regardless of matching lines. Exits with a 0 status if an input line is selected.
-s	Displays only error messages. This is useful for checking status.
-v	Displays all lines except those that match the specified pattern.
-w	Does a word search.
-x	Displays lines that match the specified pattern exactly with no additional characters.
-y	Ignores the case of letters when making comparisons.

Exit Status

This command returns the following exit values:

0	A match was found.
1	No match was found.

>1 A syntax error was found or a file was inaccessible (even if matches were found).

Examples

To use an extended pattern that contains some of the pattern-matching characters +, ?, |, (, and), enter:

```
egrep "\(([A-z]+|[0-9]+)\)" my.txt
```

This displays lines that contain letters in parentheses or digits in parentheses, but not parenthesized letter-digit combinations. It matches `(y)` and `(783902)`, but not `(alpha19c)`.

> **Note:** When using the **egrep** command, \ ((backslash followed by open parenthesis) or \ ((backslash followed by close parenthesis) match parentheses in the text, but ((open parenthesis) and) (closed parenthesis) are special characters that group parts of the pattern. The reverse is true when using the **grep** command.

Files

/usr/bin/egrep	Contains the hard link to the **egrep** command.
/bin/egrep	Specifies the symbolic link to the **egrep** command.

Related Information

The **awk** command, **ed** command, **fgrep** command, **grep** command, **sed** command.

Files in *Operating system and device management*.

Input and output redirection in *Operating system and device management*.

Shells in *Operating system and device management*.

National Language Support Overview in *AIX Version 6.1 National Language Support Guide and Reference*.

---

eimadmin Command

Purpose

Manages Enterprise Identity Mapping (EIM) domains.

Syntax

eimadmin -a | -p | -l | -m | -e -D | -R | -I | -A | -C [-s *switch*] **[-v** *verboseLevel*] **[-c** *accessType*] **[-f** *accessUserType*] **[-g** *registryParent*] **[-i** *identifier*] **[-j** *otherIdentifier*] **[-k** *URI*] **[-n** *description*] **[-o** *information*] **[-q** *accessUser*] **[-r** *registryName*] **[-t** *associationType*] **[-u** *registryUser*] **[-x** *registryAlias*] **[-y** *registryType*] **[-z** *registryAliasType*] **[-d** *domainDN*] **[-h** *ldapHost*] **[-b** *bindDN*] **[-w** *bindPassword*] **[-K** *keyFile* **[-P** *keyFilePassword*] **[-N** *certificateLabel*]] **[-S** *connectType*]

Description

The **eimadmin** command is an AIX System Services Shell tool. An administrator can use it to define an EIM domain and prime the domain with registries, identifiers, and associations between identifiers and registry users. An administrator can also use **eimadmin** to give users (and other administrators) access to an EIM domain, or list or remove the EIM entities.

Administrators can use the **eimadmin** command in two ways:

- By including information with command-line options on an **eimadmin** command
- By including information in an input file that an **eimadmin** command references

You can create the file manually or by exporting records from a database. The administrator directs utility processing by specifying a combination of command-line options.

The **eimadmin** command can perform the following actions:
- Add an object (**-a**)
- Purge an object (**-d**)
- List objects (**-l**)
- Modify attributes associated with objects (**-m**)
- Erase attributes (**-e**)

on the following objects:
- Domains (**-D**)
- Registries (**-R**)
- Identifiers (**-I**)
- Associations (**-A**)
- Access authorities (**-C**)

Notes:
1. Each **eimadmin** command must include one action and one object type. Depending on the object and the action you are performing on it, EIM might require additional parameters.
2. Some options are for multivalue attributes, which you can specify more than once. Other options are for single-value attributes, which you can specify only once. (If you repeat an option that is for a single-value attribute, **eimadmin** processes only the first value it encounters in the command.) Apart from these stipulations, the order in which you specify parameters is not important.
3. You can code the parameters of the **eimadmin** command in several ways:
   - Concatenate an action and an object, omitting the embedded hyphen: `-aD`
   - Include both hyphens, and separate the two options with a space: `-a -D`

   In other words, the following example is *not* valid because it includes both hyphens and there is no space before **-D**: `-a-D`

Flags

The **eimadmin** command takes the following action flags.

-a	Adds an object. (Creates an object definition and its attributes.)
-e	Erases an attribute. (Clears a single-value attribute or removes a multivalue attribute.)
-l	Lists an object. (Retrieves an object definition and its attributes.)
-m	Modifies an attribute. (Alters an attribute of an existing object, either by changing a single-value attribute or adding a multivalue attribute.)
-p	Purges an object. (Removes an object definition and its attributes.)

The **eimadmin** command takes the following object flags.

-A	An association. This is a relationship between an identifier in the EIM domain and a user ID.
-C	An access authority. This is an EIM-defined LDAP access control group.
-D	A domain. This is a collection of identifiers, user registries, and associations between identifiers and user IDs, stored within an LDAP directory.
-I	An identifier. This is the name of a person or entity participating in an EIM domain.
-R	A registry. This is the name of a user registry. Associations are defined between identifiers and user IDs in the user registry.

The **eimadmin** command takes the following processing control flags.

-s *switch*

The *switch* specifies a value that affects the way the **eimadmin** command functions operate. You can specify the following value:

RMDEPS
Removes dependents when removing a domain or system registry. This makes it easier to remove a domain by first removing all identifiers and registries defined for the domain. It also makes it easier to remove a system registry by first removing all application registries defined for the registry.

Attention: **Attention:** The **eimadmin** command does not warn you that dependents exist before removing them, so use this switch carefully.

-v *verboseLevel*

The *verboseLevel* parameter is an integer from 1 to 10 that controls the amount of trace detail that the **eimadmin** command displays. (It is for diagnosing problems in the **eimadmin** utility.) The default value of 0 indicates no trace information. You can specify an integer value from 1 to 10, from the least to greatest amount of trace information. The utility checks the value and displays trace information defined for the level and all lower levels. The following levels trigger specific information:

- 3–indicates EIM API call parameters and return values
- 6–indicates option values and input file labels
- 9–indicates utility routine entry and exit statements

The **eimadmin** command takes the required and optional attribute flags listed in the following table. The flag options are single-valued unless otherwise indicated. If you specify an option more than once, the utility processes only the first occurrence.

Notes:

1. You can specify these attributes as command options or as fields in input files. If you are specifying command options, you must enclose values with imbedded blanks within quotation marks (″) or (’). Quotation marks are optional for single-word values. Specifying a multiword value without quotation marks in effect truncates the command line options; values after the first word are truncated.
2. The following special characters are not allowed in *registryName*, *registryParent*, or *identifier*:
   `, = + < > # ; \ *`

-c *accessType*

Specifies the scope of access authority the user has over the EIM domain. *accessType* must be one of the following values:

ADMIN Specifies administrative access.

REGISTRY
Specifies registry access. If you specify **REGISTRY**, you must also specify a registry value (**-r**). The registry value can be a specific registry name or it can be an asterisk (*) to indicate access to all registries.

IDENTIFIER
Specifies identifier access.

MAPPING
Specifies mapping operations access.

-f *accessUserType*

Specifies the type for the access user name. *accessUserType* must be one of the following types:

DN The *accessUser* is a distinguished name.

KERBEROS
The *accessUser* is a Kerberos identity.

-g *registryParent* Specifies the name of a system registry. An application registry is a subset of a system registry. If you are adding an application registry, you must use the **-r** option and the **-g** option. The **-r** value is the application registry you are defining. The **-g** option is the preexisting system registry.

-i *identifier* Specifies a unique identifier name. For example: `John Day`.

-j *otherIdentifier* Specifies a nonunique identifier name. For example: `John`.
Note: You can specify this option multiple times to assign multiple nonunique identifiers.

-k *URI* Specifies the Universal Resource Identifier (URI) for the registry (if one exists).

-n *description* Specifies any text (that you provide) to associate with the domain, registry, identifier, or association.
Note: You can define a user description only for target associations.

-o *information* Specifies additional information to associate with an identifier or association.
Note: You can define user information only for target associations. You can specify this option multiple times to assign multiple pieces of information.

-q *accessUser* Specifies the user distinguished name (DN) or the Kerberos identity with EIM access, depending on the *accessUserType* specified.

-r *registryName* Specifies the name of a registry. When you add a new registry, **eimadmin** treats the registry as a system registry unless you also specify the **-g** option. If you specify the **-g** option, **eimadmin** treats the registry as an application registry.

-t *associationType* Specifies the relationship between an identifier and a registry. *associationType* must be one of the following:

ADMIN Indicates associating a user ID with an identifier for administrative purposes.

SOURCE
Indicates that the user ID is the source (or from) of a lookup operation.

TARGET
Indicates that the user ID is the target (or to) of a lookup operation.

Note: You can specify this option multiple times to define multiple relationships.

-u *registryUser* Specifies the user ID of the user defined in the registry.

-x *registryAlias* Specifies another name for a registry. You must specify this option multiple times to assign multiple aliases.

-y *registryType* Specifies the type of registry. Predefined types that **eimadmin** recognizes include the following:

- RACF
- OS/400®
- KERBEROS (for case ignore)
- KERBEROSX (for case exact)
- AIX
- NDS
- LDAP
- PD (Policy Director)
- WIN2K

You can also create your own types by concatenating a unique OID with one of the following two normalization methods:

- caseIgnore
- caseExact

-z *registryAliasType*
Specifies the type for a registry alias. You can invent your own value or use one of the following suggested values:

- DNSHostName
- KerberosRealm
- IssuerDN
- RootDN
- TCPIPAddress
- LdapDnsHostName

Note: For a set of command line options or single input data record, the **eimadmin** command recognizes only the first specification of *registryAliasType*. However, the **eimadmin** command does recognize multiple registry aliases and associates all of them with the single *registryAliasType*.

The **eimadmin** command takes the following connection type flags.

-b *bindDN*
Specifies the distinguished name to use for the simple bind to LDAP.

-d *domainDN*
Specifies the full distinguished name (DN) of the EIM domain. *domainDN* begins with `'ibm-eimDomainName='` and consists of the following elements:

domainName
The name of the EIM domain you are creating. For example, **MyDomain**.

parent distinguished name
The distinguished name for the entry immediately above the given entry in the directory information tree hierarchy, such as `o=ibm,c=us`. For example:

```
ibm-eimDomainName=MyDomain,o=ibm,c=us
```

-h *ldapHost*
Specifies the URL and port for the LDAP server controlling the EIM data. The format is:

```
ldap://some.ldap.host:389
ldaps://secure.ldap.host:636
```

-K *keyFile*
Specifies the name of the SSL key database file, including the full path name. If the file cannot be found, it is assumed to be the name of a RACF key ring that contains authentication certificates. This value is required for SSL communications with a secure LDAP host (prefixed `ldaps://`). For example:

```
/u/eimuser/ldap.kdb
```

-N *certificateLabel*
Specifies which certificate to use from the key database file or RACF key ring. If this option is not specified, the certificate marked as the default in the file or ring is used.

-P *keyFilePassword*
Specifies the password required to access the encrypted information in the key database file. Alternatively, you can specify an SSL password stash file for this option by prefixing the stash file name with `file://`. For example:

```
secret  or file:///u/eimuser/ldapclient.sth
```

Note: The **eimadmin** command prompts for a key file password if you specify the name of a key database file for the **-K** option but not the **-P** option on the command line.

-S *connectType*
Specifies the method of authentication to the LDAP server. *connectType* must be one of the following values:

- **SIMPLE** (bind DN and password)
- **CRAM-MD5** (bind DN and protected password)
- **EXTERNAL** (digital certificate)
- **GSSAPI** (Kerberos)

If not specified, *connectType* defaults to **SIMPLE**. For connect type **GSSAPI**, the default Kerberos credential is used. This credential must be established using a service such as kinit prior to running **eimadmin**. For KINIT and related information, refer to the AIX Authentication Service Administration.

-w *bindPassword* Specifies the password associated with the bind DN.

The connection information needed by the utility includes the EIM domain (**-d**) and its controlling server (**-h**), the identity (**-b**,**-w**; or **-K**,**-P**,**-N**) with which to authenticate (bind) to the server, and the authentication method (**-S**).

For object types other than domain (**-D**), specifying the domain, server and bind identity is optional. If these are not specified, the information is retrieved from a RACF profile.

Note: If any of the connect information is specified, the full set of values required for the connect type must also be specified. Omitting one or more values (but not all) results in an error. The following table shows the required and optional values for each connect and host type when specified with the **eimadmin** command.

Connection Type/Host Type	Required Values	Optional Values
SIMPLE or CRAM-MD5/secure (`ldaps://`)	-d, -h, -b, -w, -K, -P	-N
SIMPLE or CRAM-MD5/nonsecure (`ldap://`)	-d, -h, -b, -w	
EXTERNAL/secure (`ldaps://`)	-d, -h, -K, -P, -S	-N
EXTERNAL/nonsecure (`ldap://`)	unsupported	unsupported
GSSAPI/secure (`ldaps://`)	-d, -h, -K, -P, -S	-N
GSSAPI/nonsecure (`ldap://`)	-d, -h, -S	

Notes:

1. There are two exceptions to the preceding table:
   - The domain option (**-d**) is not required for domain functions if the value is specified through an input file.
   - An SSL key database file password or stash file (-P) is not required when -K specifies a RACF key ring.
2. The **eimadmin** command prompts for the simple bind password if it is required and **-w** is not specified on the command line, and prompts for the SSL key database file password if it is required and **-P** is not specified on the command line.

The following table summarizes required and optional flags for each object type and action pair. You can specify the value for most options in an input file instead of specifying it on the command line.

Object Type (Action)	Flags	Comments
D (**a**)	• Required: **d**, **h** • Optional: **n**	Add a domain.
D (**p**)	• Required: **d**, **h** • Optional: **s**	Remove a domain. If the domain is not empty, include `-s RMDEPS`.
D (**l**)	• Required: **d**, **h** • Optional:	List domains. Specify `-d*` to list all domains.
D (**m**)	• Required: **d**, **h** • Optional: **n**	Modify or add a domain attribute.
D (**e**)	• Required: **d**, **h** • Optional: **n**	Remove or clear a domain attribute.

Object Type (Action)	Flags	Comments
R (**a**)	• Required: **r**, **y** • Optional: **g**, **k**, **n**, **x**, **z**	Add a registry. The value specified for **-r** is assumed to be a new system registry unless **-g** is also specified, in which case the **-r** value indicates a new application registry.
R (**p**)	• Required: **r** • Optional: **s**	Remove a registry.
R (**l**)	• Required: **r** • Optional: **y**	List registries. Return all registry entries in the domain that match the specified **-r** value search filter, which might contain the wild card *****.
R (**m**)	• Required: **r** • Optional: **k**, **n**, **x**, **z**	Modify or add a registry attribute, including a registry alias.
R (**e**)	• Required: **r** • Optional: **k**, **n**, **x**, **z**	Remove or clear a registry attribute, including a registry alias.
I (**a**)	• Required: **i** • Optional: **j**, **n**, **o**	Add an identifier.
I (**p**)	• Required: **i** • Optional:	Remove an identifier.
I (**l**)	• Required: **i** • Optional:	List an identifier by unique identifier name. Return all identifier entries in the domain that matches the specified **-i** value search filter, which might contain the wild card *****.
I (**l**)	• Required: **j** • Optional:	List an identifier by nonunique identifier name. Return all identifier entries in the domain that have a nonunique identifier matching the specified **-j** value search filter, which might contain the wild card *.
I (**m**)	• Required: **i** • Optional: **j**, **n**, **o**	Modify or add an identifier attribute.
I (**e**)	• Required: **i** • Optional: **j**, **n**, **o**	Remove or clear an identifier attribute.
A (**a**)	• Required: **i**, **r**, **u**, **t** • Optional: **n**, **o**	Add an association. You can repeat the **-t** option to add multiple associations types. The **-n** and **-o** flags are relevant only to TARGET associations.
A (**p**)	• Required: **i**, **r**, **u**, **t** • Optional:	Remove an association. You can repeat the **-t** option to remove multiple associations types.
A (**l**)	• Required: **i** • Optional: **t**	List associations. Return all associations in the domain for specified **-i** unique identifier. Specify a **-t** value to limit the entries returned to the given association type.
A (**m**)	• Required: **r**, **u** • Optional: **n**, **o**	Modify or add an association attribute. The **-n** and **-o** flags are relevant only to TARGET associations.
A (**e**)	• Required: **r**, **u** • Optional: **n**, **o**	Remove or clear an association attribute. The **-n** and **-o** flags are relevant only to TARGET associations.
C (**a**)	• Required: **c**, **q**, **f** • Optional: **r**	Add access. For access type REGISTRY, provide a specific **-r** registry value, or a wild card * indicating access to all registries in the domain.
C (**p**)	• Required: **c**, **q**, **f** • Optional: **r**	Remove access. For access type REGISTRY, provide a specific **-r** registry value, or a wild card * indicating access to all registries in the domain.

Object Type (Action)	Flags	Comments
C (**l**)	• Required: **c** • Optional: **r**	List access by type. For access type REGISTRY, provide a specific **-r** registry value, or a wild card ***** indicating access to all registries in the domain.
C (**l**)	• Required: **q**, **f** • Optional:	List access by user.

Exit Status

The **eimadmin** command returns one of the following exit codes upon completion:

0 Successful.
4 One or more errors encountered but, if you specified an input file, all records were processed.
8 A severe error occurred that caused processing to stop before reaching the end of an input file, if specified.

Examples

1. To list a single domain, type:

   ```
   eimadmin -lD -h ldap://my.server -b "cn=EIM admin,o=MyCompany,c=US" -d "ibm-eimDomainName=My Employees,o=My Company,c=US"
   ```

   This returns something similar to the following output:

   ```
   domain name: My Employees
   domain DN: ibm-eimDomainName=My Employees,o=My Company,c=US
   description: employees in my company
   ```

2. To list a single registry, type:

   ```
   eimadmin -lR -r MyRegistry
   ```

   This returns something similar to the following output:

   ```
   registry: MyRegistry
   registry kind: APPLICATION
   registry parent: MySystemRegistry
   registry type: RACF
   description: my racf registry
   URI: ldap://some.big.host:389/profileType=User,cn=RACFA,o=My Company,c=US
   registry alias: TCPGROUP
   registry alias type: DNSHostName
   ```

3. To list identifiers, type:

   ```
   eimadmin -lI -i "J.C.Smith"
   ```

   This returns something similar to the following output:

   ```
   unique identifier: J.C.Smith
   other identifier: J.C.Smith
   other identifier: Joseph
   other identifier: Joe
   description: 004321
   information: D01
   information: 1990-04-11
   ```

4. To list target associations, type:

   ```
   eimadmin -lA -i "J.C.Smith" -t target
   ```

   This returns something similar to the following output:

   ```
   unique identifier: J.C.Smith
   registry: MyRegistry
   registry type: RACF
   ```

```
association: target
registry user: SMITH
description: TSO
information: 1989-08-01
information: ADMIN1
```

5. To list accesses, type:

```
eimadmin -lC -c admin
```

This returns something similar to the following output:

```
access user: cn=JoeUser,o=My Company,c=us
access user: cn=admin1,o=My Company,c=us
access user: cn=admin2,o=My Company,c=us
```

Location

/usr/bin/eimadmin

Security

The LDAP administrator has the authority to use the **eimadmin** command and access to all the functions it provides. EIM administrators can use the command as long as the following conditions are true:

- They have a bind distinguished name and password defined at the LDAP server containing the EIM domain
- Their bind distinguished name has one of the EIM authorities:
  - EIM administrator
  - EIM registries administrator
  - EIM registry X administrator
  - EIM identifiers administrator

Standard Error

The **eimadmin** command issues a message to prompt for a password or to indicate an error. Do not expect to receive a message for successful completion unless you use an input file. When processing records in an input file, **eimadmin** issues an informational message as the process starts and stops, in addition to a progress message every 50 records.

Note: The **eimadmin** command returns one or more data lines for list (-l) requests unless it finds no matching EIM entries, or the bind identity is not authorized to access that data.

Related Information

The **eimadmin.conf** file.

---

elogevent Command

Purpose

Logs event information generated by the event response resource manager (ERRM) to a specified log file.

Syntax

elogevent [**-h**] *log_file*

Description

The **elogevent** captures event information that is posted by the event response resource manager (ERRM) in environment variables the ERRM generates when an event occurs. This script can be used as

an action that is run by an event response resource. It can also be used as a template to create other user-defined actions. This script always return messages in English.

Event information that is returned about the ERRM environment variables includes the following:

Local Time
Time when the event or rearm event is observed. The actual environment variable supplied by ERRM is ERRM_TIME. This value is localized and converted to readable form before being displayed.

This script uses the **alog** command to write event information to and read event information from the specified *log_file*.

Flags

–h Writes the script's usage statement to standard output.

Parameters

log_file
Specifies the name of the file where event information is logged. An absolute path for the *log_file* parameter should be specified.

The *log_file* is treated as a circular log and has a fixed size of 64KB. When *log_file* is full, new entries are written over the oldest existing entries.

If *log_file* already exists, event information is appended to it. If *log_file* does not exist, it is created so that event information can be written to it.

Exit Status

0 The script has run successfully.

1 A required *log_file* is not specified.

2 The *log_file* path is not valid.

Restrictions

- This script must be run on the node where the ERRM is running.
- The user who runs this script must have write permission for the *log_file* where the event information is logged.

Standard Output

When the **-h** flag is specified, the script's usage statement is written to standard output.

Examples

1. To log information, specify **/tmp/event.log** in the Web-based System Manager interface. ERRM runs this command:

   ```
   /usr/sbin/rsct/bin/elogevent/tmp/event.log
   ```

   The **/tmp/event.log** file does not need to exist when the command is run.
2. To see the contents of the **/tmp/event.log** file, run this command:

   ```
   alog -f /tmp/event.log -o
   ```

   The following sample output shows a warning event for the **/var** file system (a file system resource):

```
================================================================================
Event reported at Mon Mar 27 16:38:03 2007

Condition Name:                         /var space used
Severity:                               Warning
Event Type:                             Event
Expression:                             PercentTotUsed>90

Resource Name:                          /var
Resource Class Name:                    IBM.FileSystem
Data Type:                              CT_UINT32
Data Value:                             91
```

Location

/usr/sbin/rsct/bin/elogevent

Related Information

Commands: **alog**, **logevent**

---

emgr Command

Purpose

Starts the interim fix (interim fix) manager, which installs, removes, lists, and checks system interim fixes.

Syntax

emgr -l [**-L** *Label* | **-n** *interim fixNumber* | **-u** *VUID*] [**-v**{**1**|**2**|**3**}] [**-X**] [**-a** *path*]

emgr -e *interim fixPackage* | **-f** *ListFile* [**-w** *Directory*] [**-b**] [**-k**] [**-p**] [**-l**] [**-q**] [**-m**] [**-o**] [**-X**] [**-a** *path*]

emgr -i *interim fixPackage* | **-f** *ListFile* [**-w** *Directory*] [**-b**] [**-C**] [**-k**] [**-p**] [**-l**] [**-q**] [**-X**] [**-a** *path*]

emgr -C -L *Label* [**-b**] [**-k**] [**-p**] [**-l**] [**-q**] [**-X**]

emgr -r -L *Label* | **-n** *interim fixNumber* | **-u** *VUID* | **-f** *ListFile* [**-w** *Directory*] [**-a** *path*] [**-b**] [**-k**] [**-p**] [**-l**] [**-q**] [**-X**]

emgr -c [**-L** *Label* | **-n** *interim fixNumber* | **-u** *VUID* | **-f** *ListFile*] [**-w** *Directory*] [**-a** *path*] [**-v**{**1**|**2**|**3**}] [**-X**]

emgr -M | **-U** [**-L** *Label* | **-n** *interim fixNumber* | **-u** *VUID* | **-f** *ListFile*] [**-w** *Directory*] [**-a** *path*] [**-X**]

emgr -R *interim fixLabel* [**-w** *Directory*] [**-a** *path*] [**-X**]

emgr -P [*Package*] [**-a** *path*] [**-X**]

emgr -d -e *interim fixPackage* | **-f** *ListFile* [**-w** *Directory*] [**-v**{**1**|**2**|**3**}]

Description

The **emgr** (interim fix manager) command can be used to install and manage system interim fixes. The interim fix manager installs packages created with the **epkg** command and maintains a database containing interim fix information. The **emgr** command performs the following operations:

- interim fix package installation
- interim fix removal
- interim fix listing

- interim fix checking
- interim fix mounting
- interim fix unmounting
- Display package locks
- Force removal of an installed interim fix

Referencing an Efix

The ways to reference an interim fix are as follows:

Reference by Label

Each interim fix that is installed on a given system will have a unique interim fix label. This is the unique key that binds all of the different database objects. To reference an interim fix by label, pass the label as a parameter to the **-L** flag. For example, to run a check operation on an interim fix with label **ABC123**, enter:

```
emgr -cL ABC123
```

Reference by Efix ID

Each interim fix that is installed on a given system has an interim fix ID. The interim fix ID is simply the order number in which the interim fix is listed in the interim fix database. Using this option may be convenient if you are performing operations on interim fixes based on interim fix listings. The **emgr** command will convert the interim fix ID into an interim fix label before performing the given operation. To reference an interim fix by ID, pass the ID as an parameter to the **-n** flag.

Note: Efix IDs can change as interim fixes are removed and added. Always verify the current interim fix ID number by using the **-l** flag to list the specific interim fix or all interim fixes.

For example, to run a check operation on the first interim fix with ID equal to 1, enter:

```
emgr -cn1
```

Reference by VUID

Because interim fix packages are not formally tracked by any entity, it is possible that the same interim fix label could be used for more than one interim fix package. However, the **emgr** command does not accept the installation of more than one interim fix with the same interim fix label at the same time. The VUID (Virtually Unique ID) can be used to differentiate packages with the same interim fix label. The **emgr** command converts the VUID into an interim fix label before performing the given operation. For example, to list an installed interim fix with VUID equal to **000775364C00020316020703**, enter:

```
emgr -l -u 000775364C00020316020703
```

Note: The VUID is displayed in the preview phase of interim fix installation and removal. The VUID is also displayed when listing with verbosity level set to 2 or higher with the **-v** flag.

Efix Logging

The following operations are logged to the **emgr** command log file, **/var/adm/ras/emgr.log**:

- Installation
- Removal
- Checking
- Mounting
- Unmounting
- Forced Removal

Concurrent Updates

The **emgr** command supports the installation of a new kind of interim fix called a concurrent update. This form of interim fix contains a modification to the AIX kernel, or one of its kernel extensions, that can be applied directly to the system memory and does not require the system to be rebooted. This direct

patching to the system memory allows you to safely evaluate and test a kernel modification without modifying the file containing the system's current kernel on the disk. Any concurrent update applied to the system memory will not persist after a system reboot unless you choose to commit the changes introduced by the concurrent update to the disk using the **-C** flag.

Flags

-a *path*
Specifies an alternate directory path for installation.

-b
Causes the **emgr** command to skip the usual AIX **bosboot** process for interim fixes that require rebooting.

-c
Specifies the check operation. Instructs the **emgr** command to run a check operation on the specified interim fix or interim fixes.

-C
Commits an interim fix containing concurrent updates to the disk. This option must be used along with the **-i** option, or can be used after an interim fix has been applied with the **-i** option. This causes the concurrent updates to persist across system reboots.

-d
Displays the contents and topology. This option is useful with the **-v** flag in displaying verbosity output.

-e *interim fixPackage*
Specifies the path of the interim fix package file. The interim fix package file must be created with the **epkg** command and must end with the 16-bit compression extension, **.Z**.

-f *ListFile*
Specifies a file that contains one of the following:

- A list of package locations for the installation operation (one per line)
- A list of interim fix labels for the remove, mount, unmount, and check operations (one per line)

The **emgr** command ignores any blank lines or lines where the first non-white-space character is the **#** character.

-i *interim fixPackage*
Specifies the path of an interim fix package file containing a concurrent update, and applies the concurrent update to the system memory. The update does not persist across system reboots unless the **-C** flag is used.

-l
Runs the low-level debugger for AIX **bosboot** by using the **bosboot** command's **-l** flag.

-k
Loads the low-level debugger during AIX **bosboot** using the **bosboot** command's **-D** flag.

-l
Instructs the **emgr** command to run the list operation on the specified interim fix or interim fixes.

-L *Label*
Selects the interim fix for this operation by interim fix label.

-m
Instructs the **emgr** command to perform a mount installation. When and interim fix is mount-installed, the interim fix files are mounted over the target files.

-M
Instructs the **emgr** command to mount an interim fix or interim fixes that have been mount-installed by using the **-m** flag. The **-M** flag can be used to mount an interim fix that was installed using the **-m** flag and has been unmounted by the **-U** flag or by some other means, such as rebooting the system.

-n *interim fixID*
Selects the interim fix for this operation by specifying the interim fix ID.

-o
Specifies that the interim fix installation can overwrite an existing package.

-p
Instructs the **emgr** command to perform a preview for either installation or removal. The preview runs all of the check operations, but does not make any changes.

-P [*Package*]
Specifies the package-view operation, which displays all packages that are locked by the interim fix manager, their installer, and the locking label or labels.

-q
Suppresses all output other than errors and strong warnings.

-r
Instructs the **emgr** command to run a remove operation on the specified interim fix or interim fixes.

-R *Label*
Instructs the **emgr** command to run a force-remove operation. This option removes interim fix data and package locks associated with the interim fix label without actually removing interim fix files, running any remove scripts, or boot processing. This option can be used for only one interim fix at a time. The interim fix label is required to identify the target interim fix.

Attention: This method of interim fix removal should be considered an emergency procedure. Because this method can create inconsistencies on the target system, the force remove method should be used only if all other methods of removing the interim fix are unsuccessful.

-u *VUID*
Selects the interim fix for this operation by specifying the VUID.

-U
Instructs the **emgr** command to unmount an interim fix or interim fixes that have been mount-installed by using the **-m** flag.

-v{**1**|**2**|**3**} Specifies the verbosity level for the listing operation or the verification level for the check operation. Valid levels are 1, 2, and 3.

-w *Directory* Instructs the **emgr** command to use the specified working directory instead of the default **/tmp** directory.

-X Attempts to expand any file systems where there is insufficient space to perform the requested **emgr** operation. This option expands file systems based on available space and size estimates that are provided by the interim fix package and the **emgr** command.

Notes:

1. It is possible to exhaust available disk space during an installation even if the **-X** flag is used. This is more likely if other files are being created or expanded in the same file systems during an installation.
2. Remote file systems cannot be expanded by the **emgr** command.

Exit Status

0 All of the **emgr** command operations completed successfully.

>0 An error occurred.

Security

Only the root user can run the **emgr** command. Efix data, saved files, and temporary files are accessible only by the root user.

The **emgr** command looks for a supported MD5 generating command on the system. If one is located, the **emgr** command displays the MD5 checksum to the user. The user can then cross check this MD5 sum with a secured source. If an MD5 generating command is not located, the **emgr** command takes no further action.

The user can force set the path to an MD5 command by exporting the EMGR_MD5_CMD shell variable. This variable should contain the absolute path to the MD5 generating command.

Note: This feature is not supported in the original release of interim fix management. It is recommended that the user updates to the latest level of interim fix management by updating **bos.rte.install** to the latest level.

Examples

1. To preview the installation of an interim fix package called **games.020303.epkg.Z**, enter:
   ```
   emgr -p -e games.020303.epkg.Z
   ```
2. To install the interim fix package called **games.020303.epkg.Z** and automatically expand file systems if additional space is needed, enter:
   ```
   emgr -X -e games.020303.epkg.Z
   ```
3. To list all interim fixes on the system, enter:
   ```
   emgr -l
   ```
4. To do a level 3 listing of interim fix label **games**, enter:
   ```
   emgr -lv3 -L games
   ```
5. To remove the interim fix with label **games**, enter:
   ```
   emgr -r -L games
   ```
6. To preview the removal of the interim fix labels in file **/tmp/myfixes**, enter:
   ```
   emgr -rp -f /tmp/myfixes
   ```
7. To check all interim fixes with verification level 2, enter:
   ```
   emgr -cv2
   ```

8. To check interim fix ID number 3 with verification level 1 (the default verification level), enter:
   ```
   emgr -c -n3
   ```
9. To check interim fix with VUID of **000775364C00020316020703** and verification level 3, enter:
   ```
   emgr -u 000775364C00020316020703 -c -v3
   ```
10. To list all locked packages and their interim fix labels, enter:
    ```
    emgr -P
    ```
11. To list all interim fix labels that have locked the **installp** package **bos.rte.lvm**, enter:
    ```
    emgr -P bos.rte.lvm
    ```
12. To mount-install the interim fix package called **games.020303.epkg.Z** and suppress AIX **bosboot**, enter:
    ```
    emgr -e games.020303.epkg.Z -mb
    ```
13. To mount all interim fix files that have been mount-installed on the system by using the **-m** option, enter:
    ```
    emgr -M
    ```
14. To unmount all interim fix files associated with interim fix label **games**, enter:
    ```
    emgr -U -L games
    ```
15. To apply an interim fix package called **kernelmod.031007.epkg.Z** with concurrent updates to the system memory, enter:
    ```
    emgr -i kernelmod.031007.epkg.Z
    ```
16. To commit the concurrent updates associated with the interim fix label **kernelmod** to the disk, enter:
    ```
    emgr -C -L kernelmod
    ```
17. To apply an interim fix package called **kernelmod2.031007.epkg.Z** with concurrent updates to the system memory, and also to commit the concurrent updates to the disk, enter:
    ```
    emgr -i kernelmod2.031007.epkg.Z -C
    ```
18. To display level 3 verbosity output on interim fix package **test.102403.epkg.Z**, enter:
    ```
    emgr -v3 -d test.102403.epkg.Z
    ```

Files

/usr/sbin/emgr	Contains the **emgr** command
/usr/emgrdata/DBS/efix.db	Contains the interim fix header database
/usr/emgrdata/DBS/files.db	Contains the interim fix files database
/usr/emgrdata/DBS/pkglck.db	Contains the package locks database
/usr/emgrdata/DBS/prereq.db	Contains the prerequisite database
/usr/emgrdata/DBS/e2eprereq.db	Contains the interim fix prerequisite database

Related Information

The **bosboot** command, **epkg** command.

Installing optional software products and service updates in *Installation and migration*.

---

emstat Command

Purpose

Shows emulation exception statistics.

Syntax

emstat [**-a** | **-v**] [*Interval*] [*Count*]

Description

The **emstat** command displays emulation exception statistics. Emulation exceptions can occur when some legacy applications or libraries, which contain instructions that have been deleted from older processor architectures, are executed on newer processors. These instructions may cause illegal instruction program exceptions. The operating system catches these exceptions and emulates the older instruction(s) to maintain program functionality, potentially at the expense of program performance.

The emulation exception count since the last time the machine was rebooted and the count in the current interval are displayed. The user can optionally display alignment exception statistics or individual processor emulation statistics.

The default output displays statistics every second. The sampling interval and number of iterations can also be specified.

Parameters

Interval	Interval between samples.
Count	Number of iterations.

Flags

-a	Displays alignment exception statistics. This flag cannot be used with the **-v** flag.
-v	Display individual processor statistics. This flag cannot be used with the **-a** flag.

Examples

1. To display the emulation statistics every second, type:

   ```
   emstat
   ```

   This produces the following output:

   ```
     Emulation  Emulation
     SinceBoot      Delta
       8845591          0
       8845591          0
       8845591          0
       8845591          0
       8845591          0
       8845591          0
     ...
   ```

2. To display emulation and alignment exception statistics every two seconds, a total of 5 times, type:

   ```
   emstat -a 2 5
   ```

   This produces the following output:

   ```
     Alignment  Alignment  Emulation  Emulation
     SinceBoot      Delta  SinceBoot      Delta
      21260604          0   70091846          0
      23423104    2162500   72193861    2102015
      25609796    2186692   74292759    2098898
      27772897    2163101   76392234    2099475
      29958509    2185612   78490284    2098050
   ```

3. To display emulation statistics, every 5 seconds, for each processor, type:

   ```
   emstat -v 5
   ```

   This produces the following output:

```
Emulation  Emulation  Emulation  Emulation
SinceBoot      Delta    Delta00    Delta01
 88406295          0          0          0
 93697825    5291530          0    5291530
 98930330    5232505    5232505          0
102595591    3665261     232697    3432564
102595591          0          0          0
```

Related Information

The **alstat** command.

---

emsvcsctrl Command

Purpose

Starts the event management subsystem.

Syntax

emsvcsctrl [**–a** | **–s** | **–k** | **–d** | **–c** | **–t** | **–o** | **–h**]

Description

emsvcsctrl is a control script that starts the event management subsystem. Event management is a distributed subsystem of RSCT that provides a set of high-availability services for the IBM RS/6000 server. 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 system. The emsvcsctrl 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 emsvcs. A daemon is associated with each subsystem. The emsvcsctrl script also controls the operation of the AIX Resource Monitor subsystem. The subsystem is under SRC control and also belongs to the emsvcs subsystem group. A daemon is associated with each subsystem.

Instances of the Event Management and AIX Resource Monitor subsystems execute on each node in the HACMP/ES cluster. From an operational point of view, the Event Management subsystem group is organized as follows:

Subsystem
: Event Management

Subsystem Group
: emsvcs

SRC Subsystem
: The emsvcs subsystem is associated with the haemd daemon.

emaixos
: The emaixos is associated with the harmad daemon.

Daemons
: The haemd daemon provides the Event Management services. The harmad daemon is the resource monitor for AIX operating system resources.

The emsvcsctrl script is not normally executed from the command line. It is normally called by the HACMP/ES startup script command during installation of the system.

The emsvcsctrl script provides a variety of controls for operating the Event Management subsystem:

- Adding, starting, stopping, and deleting the subsystem

- Cleaning up the subsystems
- Turning tracing on and off

Adding the Subsystem: When the -a flag is specified, the control script uses the mkssys command to add the Event Management and AIX Resource Monitor subsystems to the SRC. The control script operates as follows:

1. It makes sure that the emsvcs and emaixos subsystems are stopped.
2. It removes the emsvcs and emaixos subsystems from the SRC (just in case they are still there).
3. It adds the emsvcs subsystem to the SRC.
4. It adds the emaixos subsystem to the SRC.
5. It adds haemrm group using the mkgroup command, if it does not already exist. Any errors that occur are written to a log file named /var/ha/log/em.mkgroup.
6. It creates the /var/ha/lck/haem and /var/ha/soc/haem directories, if they don't already exist. Any errors that occur are written to a log file named /var/ha/log/em.mkdir.
7. It copies the Event Management Configuration Database, (EMCDB) from its install location, /usr/sbin/rsct/install/config/em.HACMP.cdb to its run-time location, /etc/ha/cfg/em.HACMP.cdb. Any errors resulting from the copy are written to a log file named /var/ha/log/em.cp.

Starting the Subsystem: When the -s flag is specified, the control script uses the startsrc command to start the Event Management subsystem, emsvcs, and the AIX Resource Monitor subsystem, emaixos.

Stopping the Subsystem: When the -k flag is specified, the control script uses the stopsrc command to stop the Event Management subsystem, emsvcs, and the AIX Resource Monitor subsystem, emaixos.

Deleting the Subsystem: When the -d flag is specified, the control script uses the rmssys command to remove the Event Management and AIX Resource Monitor subsystems from the SRC. The control script operates as follows:

1. It makes sure that the emsvcs and emaixos subsystems are stopped.
2. It removes the emsvcs and emaixos subsystems from the SRC using the rmssys command.

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 by using the stopsrc -g emsvcs command.
2. It removes all instances of subsystems in the subsystem group from the SRC using the rmssys command.
3. It removes the Event Management Configuration Database (EMCDB) from its run-time location, /etc/ha/cfg/em.HACMP.cdb.

Turning Tracing On: When the -t flag is specified, the control script turns tracing on for the haemd daemon, using the haemtrcon command. Tracing for the harmad daemon is also enabled, using the traceson command.

Turning Tracing Off: When the -o flag is specified, the control script turns tracing off for the haemd daemon, using the haemtrcoff command. Tracing for the harmad daemon is also disabled, using the tracesoff command.

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.cluster_name.

Flags

-a Adds the subsystem.

-s Starts the subsystem.

-k Stops the subsystem.

-d Deletes the subsystem.

-c Cleans the subsystem.

-t Turns tracing on for the subsystem.

-o Turns tracing off for the subsystem.

-h Displays usage information.

Security

You must be running with an effective user ID of **root**.

Exit Status

0 Indicates the successful completion of the command.

1 Indicates that an error occurred.

Restrictions

This command is valid in an HACMP™ environment only.

Standard Error

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

Examples

1. To add the Event Management subsystem to the SRC, enter:
   ```
   emsvcsctrl -a
   ```
2. To start the Event Management subsystem, enter:
   ```
   emsvcsctrl -s
   ```
3. To stop the Event Management subsystem, enter:
   ```
   emsvcsctrl -k
   ```
4. To delete the Event Management subsystem from the SRC, enter:
   ```
   emsvcsctrl -d
   ```
5. To clean up the Event Management subsystem, enter:
   ```
   emsvcsctrl -c
   ```
6. To turn tracing on for the Event Management daemon, enter:
   ```
   emsvcsctrl -t
   ```
7. To turn tracing off for the Event Management daemon, enter:
   ```
   emsvcsctrl -o
   ```

Location

/usr/sbin/rsct/bin/emsvcsctrl Contains the **emsvcsctrl** script

Files

/var/ha/log/em.default.cluster_name
Contains the default log of the haemd daemon on the cluster named cluster_name.

/var/ha/log/em.cp Contains a log of any errors that occurred while copying the Event Management Configuration Database.

/var/ha/log/em.trace.cluster_name
Contains the trace log of the haemd daemon on the cluster named cluster_name.

/var/ha/log/em.mkgroup Contains a log of any errors that occurred while creating the haemrm group.

/var/ha/log/em.mkdir Contains a log of any errors that occurred while creating the /var/ha/lck/haem and /var/ha/soc/haem directories.

Related Information

Commands: **haemtrcoff**, **haemtrcon**, **lssrc**, **startsrc**, **stopsrc**

Daemons: **haemd**

---

enable Command

The **enable** command includes information for AIX Print Subsystem **enable** and the System V Print Subsystem **enable**.

AIX Print Subsystem enable Command

Purpose

Enables printer queue devices.

Syntax

enable *PrinterName ...*

Description

The **enable** command brings the printer queue devices specified by the *PrinterName* parameter on line, or enables the printer queue devices to be used with the system.

Notes:

1. You must have root user authority or belong to the printq group to run this command.
2. If you enter `enable -?`, the system displays the following error message:

```
enq: (FATAL ERROR): 0781-048: Bad queue or device name: -?
```

Security

Attention RBAC users and Trusted AIX users: This command can perform privileged operations. Only privileged users can run privileged operations. For more information about authorizations and privileges, see Privileged Command Database in *Security*. For a list of privileges and the authorizations associated with this command, see the **/etc/security/privcmds** database file.

Examples

To enable the print queue device `lp0:lpd0`, enter:

```
enable lp0:lpd0
```

Files

/etc/qconfig	Contains the queue configuration file.
/etc/qconfig.bin	Contains the digested, binary version of the **/etc/qconfig** file.
/usr/sbin/qdaemon	Contains the queuing daemon.
/var/spool/lpd/qdir/*	Contains the queue requests.
/var/spool/lpd/stat/*	Contains information on the status of the devices.
/var/spool/qdaemon/*	Contains temporary copies of enqueued files.

Related Information

The **cancel** command, **disable** command, **lp** command, **lpstat** command.

Starting and Stopping a Print Queue in *Printers and printing*.

System V Print Subsystem enable Command

Purpose

Enable LP printers

Syntax

enable ***printers***

Description

The **enable** command activates the named ***printers***, enabling them to print requests submitted by the **lp** command. If the printer is remote, the command will only enable the transfer of requests to the remote system; the **enable** command must be run again, on the remote system, to activate the printer. (Run **lpstat -p** to get the status of printers.)

When changes are made to the attributes of a print device, they are recognized by **enable**. Therefore to change the definition or allocation for a device, you must disable the printer on that device, change the device, and then run **enable**. The new device attributes will become effective when **enable** is executed.

Printer names are *system-defined words* and as such should be restricted to uppercase and lowercase ASCII characters.

If you enter `enable -?`, the system displays the command usage message and returns 0.

Security

Attention RBAC users and Trusted AIX users: This command can perform privileged operations. Only privileged users can run privileged operations. For more information about authorizations and privileges, see Privileged Command Database in *Security*. For a list of privileges and the authorizations associated with this command, see the **/etc/security/privcmds** database file.

Files

/var/spool/lp/*

References

The **lp** command, **lpstat** command.

enotifyevent Command

Purpose

Mails event information generated by the event response resource manager (ERRM) to a specified user ID.

Syntax

enotifyevent [**-h**] [*user-ID*]

Description

The **enotifyevent** script captures event information that is posted by the event response resource manager (ERRM) in environment variables the ERRM generates when an event occurs. This script can be used as an action that is run by an event response resource. It can also be used as a template to create other user-defined actions. This script always returns messages in English.

Event information that is returned about the ERRM environment variables includes the following:

Local Time
Time when the event or rearm event is observed. The actual environment variable supplied by ERRM is ERRM_TIME. This value is localized and converted to readable form before being displayed.

This script uses the **mail** command to send event information to the specified user ID. When a user ID is specified, it is assumed to be valid, and it is used without verifying it. If a user ID is not specified, the user who is running the command is used as the default.

user-ID is the optional ID of the user to whom the event information will be mailed. If *user-ID* is not specified, the user who is running the command is used as the default.

Flags

–h Writes the script's usage statement to standard output.

Parameters

log_file
Specifies the name of the file where event information is logged. An absolute path for the *log_file* parameter should be specified.

The *log_file* is treated as a circular log and has a fixed size of 64KB. When *log_file* is full, new entries are written over the oldest existing entries.

If *log_file* already exists, event information is appended to it. If *log_file* does not exist, it is created so that event information can be written to it.

Exit Status

0 Command has run successfully.

Restrictions

1. These scripts must be run on the node where the ERRM is running.
2. The **mail** command is used to read the file.

Standard Output

When the **-h** flag is specified, the script's usage statement is written to standard output.

Examples

1. Specify **user1** in Web-based System Manager to send mail to a user. The event response resource manager then runs the following command:

   ```
   /usr/sbin/rsct/bin/enotifyevent user1
   ```

2. You can use the **mail** command to read the contents of the event information. The following example shows how a warning event for the **/var** file system (a file system resource) is formatted and logged:

   ```
   ============================================================================
   Event reported at Sun Mar 26 16:38:03 2007

   Condition Name:      /var space used
   Severity:     Warning
   Event Type:      Event
   Expression:      PercentTotUsed>90

   Resource Name:     /var
   Resource Class Name:    IBM.FileSystem
   Data Type:     CT_UINT32
   Data Value:     91
   ```

Location

/usr/sbin/rsct/bin/enotifyevent

Related Information

Commands: **mail**, **notifyevent**

---

enq Command

Purpose

Enqueues a file.

Syntax

To Process a File

enq [**-**] [**-B** *CharacterPair*] [**-c**] [**-C**] [**-G**] [**-j**] [**-m** *Text*] [**-M** *File*] [**-n**] [**-N** *Number*] [**-o** *Option*] [**-P** *Queue*] [**-r**] [**-R** *Number*] [**-t** *"User"*] [**-T** *Title*] [**-Y**] [**-Z** *Name*] *File*

To Change the Priority of Print Jobs

enq -a *Number* **-#** *JobNumber*

To Display Status

enq [**-q** | **-A**] [**-L** | **-W**] [**-e**] [**-#** *JobNumber*] [**-u** *Name*] [**-w** *Seconds*] [**-s**]

To Change Queue and Queue Daemon Status

enq [**-d**] [**-D**] [**-G**] [**-K**] [**-L**] [**-q** | **-A**] [**-U**]

To Cancel Options

enq [**-X**] [**-x***Number*] [**-P***Printer*]

To Hold, Release or Move a Print Job to Another Queue

enq { **-h** | **-p** | **-Q** *NewQueue* } { **-#** *JobNumber* [**-P** *Queue*] | **-u** *User* | **-P** *Queue* }

To Queue and Hold a Print Job

enq -H *File ...*

Description

The **enq** command is a general-purpose utility for enqueuing requests to a shared resource, typically a printer device. Use the **enq** command to enqueue requests, cancel requests, alter the priority of a request, and display the status of queues and devices.

The **enq** command has five different syntax diagrams because all the flags are not meant to work together. Some of these flags are meant for file processing and accept *FileName* as an option. The other flags are used for changing the priority of a print job, displaying the status, changing the status of the queue or the queue daemon, and canceling a print job.

To enqueue files on a specific queue, use the **-P** flag (**-P** *Queue*). If more than one device services a queue, you can also request a particular device by specifying that device (*:device*) after the name of the queue. If you do not specify a device, the job is sent to the first available device. If you do not specify a file, the **enq** command copies standard input into a file and enqueues it for printing.

The **enq** command requests can have operator messages associated with them. This feature is useful in a distributed environment or on a system with many users. The messages are used to tell the printer operator such information as a request to load a special form or different color paper into the printer before allowing the job to print. These messages are specified with the **-m** and **-M** flags. The **qdaemon** command processes the **enq** command requests. When the **qdaemon** is ready to begin a request that has an associated message, the system displays the message on the console of the machine where the **qdaemon** process is running. The text of the message is accompanied by a prompt that tells the printer operator how to signal the request to continue or how to cancel the request.

The display generated by the **enq -A** command contains two entries for remote queues. The first entry contains the client's local queue and local device name and its status information. The second entry follows immediately; it contains the client's local queue name (again), followed by the remote queue name. Any jobs submitted to a remote queue are displayed first on the local side and are moved to the remote device as the job is processed on the remote machine.

Since the status commands communicate with remote machines, the status display may occasionally appear to hang while waiting for a response from the remote machine. The command will eventually time-out if a connection cannot be established between the two machines.

Notes:

1. Before you can enqueue a file, you must have read access to it. To remove a file, (see the **-r** flag) you must also have write access to the directory that contains the file.
2. If you want to continue changing the file after you issue the **enq** command but before it is printed, you must use the **-c** flag.
3. When enqueuing files on a printer, flags can be interspersed in any order.
4. The **-d** and **-G** flags are acted upon immediately. Syntax error appearing before these flags on the command line are reported. Syntax errors appearing after these flags on the command line are ignored.

Flags

File Processing Options

If you give the **enq** command a list of file names, it enqueues them all for file processing on the default device or on the specified device.

- Causes the **enq** command to act as a filter. The **enq** command automatically reads standard input if you do not specify a file or files. However, if you do specify a file, you can also use the dash (**-**) to force the **enq** command to read standard input. The dash (**-**) is actually not a flag, but a special type of file name. Therefore, it must come after all other flags have been specified on the command line.

-B *CharacterPair* Controls the printing of burst pages according to the value of *CharacterPair* as follows. (**n** = never, **a** = always, **g** = group. The first character is for header, the second character is for trailer.)

HT	Description
nn	No headers, no trailers
na	No headers, trailer on every file
ng	No header, trailer at the end of the job
an	Header on every file, no trailers
aa	Headers and trailers on every file in the job
ag	Header on every file, trailer after job
gn	Header at the beginning of job, no trailer
ga	Header at beginning of job, trailer after every file
gg	Header at beginning of job, trailer at end of job

The header and trailer stanzas in the **/etc/qconfig** file define the default treatment of burst pages.

> **Note:** In a remote print environment, the default is to print a header page and not a trailer page.

-c Copies the file. To save disk space, the **enq** command remembers the name of the file, but does not actually copy the file itself. Use the **-c** flag if you want to continue changing the file while you are waiting for the current copy to be printed.

-C Specifies that the **mail** command be used instead of the **write** command for error messages and job completion notification. (Using this flag is useful for writing PostScript® applications since it allows better feedback from the printer.) Error messages and job completion messages (both generated by the **piobe** command) and any data read from the printer are also sent back by mail.

The **-C** flag only applies to local print jobs. If you want to be notified when a job sent to a remote printer is completed, use the **-n** flag to receive a mail message.

> **Note:** There are some messages that cannot be redirected from **qdaemon** and the printer backend in any way. These are system errors and are sent directly to the /**dev**/**console** file.

-j Specifies that the message `Job number is: nnn,` where `nnn` is the assigned job number, be displayed to standard output. This occurs only if the job is submitted to a local print queue.

-m *Text* Submits an operator message with an **enq** command request. The specified text contains the message.

-M *File* Submits an operator message with an **enq** command request. The specified file contains the text of the message.

-n Notifies you when your job is finished. If the **-t** flag is also used, the **enq** command also notifies the user for whom the request is intended (see the **-t** flag).

-N *Number* Prints *Number* copies of the file. Normally, a file is printed only once.

-o *Option* Specifies that flags specific to the backend be passed to the backend. Thus, for each queue there are flags not described in this article that can be included on the **enq** command line. See the **piobe** command for a list of these flags.

-P *Queue* Specifies the queue to which the job is sent. A particular device on a queue can be specified by typing `-P Queue:Device.`

-r Removes the file after it has been successfully printed.

-R *Number* Sets the priority of the current job to *Number*. This flag is used at job submission time. Use the **-a** flag to alter priority after the job is submitted. Higher numbers assign higher priority. The default priority is 15. The maximum priority is 20 for most users and 30 for the users with root user authority.

-t *"User"* Labels the output for delivery to *User*. Normally the output is labeled for delivery to the user name of the person issuing the **enq** command request. The value of *User* must be a single word meeting the same requirements of a regular user ID.

-T *Title* Puts title on the header page and displays it when the **-q** flag is specified. Normally the job title is the name of the file. If the **enq** command reads from standard input, the job title is **STDIN.#** where **#** is the process ID of the **enq** command.

-Y Tells the **enq** command to ignore the rest of the command line after this flag. This is useful for discovering whether a queue is valid (if it is in the **/etc/qconfig** file). For example, typing `enq -P lp4 -Y` returns with an exit value of 0 if the line printer **lp4** is a valid queue; if otherwise, a nonzero value is returned. Using this flag is also good for forcing the **qdaemon** command to redigest the **/etc/qconfig** file.

-Z *Name* Specifies originator of remote print jobs.

Print Job Priority Options

-a *Number* Changes the priority of the named job to *Number*. The job must have been submitted for printing prior to entering the **enq** command with this flag. See the **-R** flag for a description of priorities. Use the **-#** flag to specify the job number. This flag is only valid for local print jobs.

-# *JobNumber* Specifies the job number used by the **enq -q** command or the **enq -a** command, and displays only the job specified in status output.

Notes:

1. Specify the **-P** *Queue* to override the default destination printer.
2. If jobs 1, 2, and 3 are in the printer queue, and you specify that you want the status of job 3 while job 1 is running, the status information will show job 1 and job 3, not only job 3.
3. If you specify a job number that does not exist, the system displays the current job number on the queue instead of an error message.

Display Status Options

-A Provides status for all queues. This is like running the **enq -q** command once for each queue in the **qconfig** file.

-e Excludes status information from queues that are not under the control of the **qdaemon** command. The status from such queues may be in different formats. The **-e** flag can be used with any combination of flags.

-L Specifies the long status. This flag can be used with the **-A** flag or the **-q** flag. This flag cannot be used with the **-W** flag. If the **-L** flag and **-W f**lag are used simultaneously, the first one specified takes precedence. Use the **-L** flag to show multiple files to be printed in a single print job.

-q Displays the status of the default queue. The **LPDEST** and **PRINTER** environment variable control the name of the default printer. If the **LPDEST** environment variable contains a value, that value is always used first. If the **LPDEST** variable has no value, the **enq** command uses the **PRINTER** environment variable. If the **PRINTER** environment variable contains no value, then the **enq** command uses the system default.

Notes:

1. Use the **-P** *Queue* flag with the **-q** flag to display the status of a particular queue.
2. Any destination command line options override both the **LPDEST** and the **PRINTER** environment variables.

-s Obtains the status of print queues without listing any files.

-u *Name* Specifies the user name for which to print job status.

-w *Seconds* Specifies continuous output of the queue status, updating the screen every *Seconds* specified until the queue is empty (see the **lpq** command). When the queue is empty, the process halts. This flag is only used with either the **-q** flag, or the **-A** flag, or the **-L** flag.

-W Specifies the wide status format with longer queue names, device names, and job numbers. Job number information is available on AIX 4.3.2 and later. This flag can be used with the **-A** flag or the **-q** flag. It cannot be used with the **-L** flag. If the **-L** flag and **-W f**lag are used simultaneously, the first one specified takes precedence.

Change the queue and queue Daemon Status Options

-d Runs the **digest** command on the **/etc/qconfig** file. Once the digest is completed, any changes to the **/etc/qconfig** file are reflected in the **/etc/qconfig.bin** file. A user must have root user authority to run this option.

In addition to the previous flags available to all users, the **enq** command accepts the following flags when they are entered by users that have root user authority. Root user authority means that you are root or you belong to the **printq** group.

Note: The following flags can only be used on local print jobs.

-D Device DOWN. Turns off the device associated with the queue. The **qdaemon** process no longer send jobs to the device, and entering the **enq -q** command shows its status as DOWN. Any job currently running on the device is allowed to finish.

-G Die GRACEFULLY. Ends the **qdaemon** process after all currently running jobs are finished. Use of this flag is the only clean way to bring the **qdaemon** process down. Use of the **kill** command may cause problems, such as jobs hanging up in the queue.

If the **qdaemon** process is running under **srcmstr** (the default configuration), **enq -G** does not prevent **qdaemon** from being restarted automatically. You must use the **chssys** command, which changes the default configuration and prevents the automatic restart of the **qdaemon** process. The following command:

```
chssys -s qdaemon -O
```

issued prior to the **enq -G** command, prevents the automatic restart of **qdaemon**.

The following command:

```
startsrc -s qdaemon
```

restarts the **qdaemon** process manually.

-K Acts the same as the **-D** flag, except that all current jobs are KILLED. They remain in the queue, and are run again when the device is turned on.

-L Specifies the long status. This flag can be used with the **-A** flag or the **-q** flag. Use the **-L** flag to show multiple files to be printed in a single print job.

-U Brings UP the device associated with a queue. The **qdaemon** process sends jobs to it again and entering the **enq -q** command shows its status as READY.

Note: If more than one device is associated with a queue, you must specify the device as well as the queue when you use the **-D** flag, the **-K** flag, and the **-U** flags. For example, entering `-P lp:lpd` designates the same device only if there is no other device on that queue.

Cancel Options

-X Cancels the printing of your jobs. If you have root user authority, all jobs on the specified queue are deleted. This flag is only valid on local print jobs.
-x *Number* Cancels the printing of the specified job *Number*.
-P *Printer* Specifies the *Printer* where either all jobs or the selected job number is to be canceled.

Attention: If you have root user authority and do not specify a queue, all jobs on all queues are deleted.

Holding and Releasing a Print Job Options

-# *JobNumber* Designates the number of the print job to be held or released.
-h Holds the specified print job.
-H Queues and holds the file indicated with the *File* parameter.
-p Releases the specified print job.
-P *Queue* Designates the print queue to be held or released.
-u *User* Designates the user whose print jobs are to be held or released.

Moving Print Job Options

-# *JobNumber* Designates the number of the print job to be moved.
-P *Queue* Designates the print queue to be moved. The value of the *Queue* variable can be a queue name or in the form queue:device name.
-Q *NewQueue* Designates the target queue where the print job will be moved to. The value of the *NewQueue* variable can be in the form of a queue name or in the form queue:device name.
-u *User* Designates the user whose print jobs are to be moved.

Security

Auditing Events:

Event	Information
ENQUE_admin	Queue name, device name, job name, user name

Attention RBAC users and Trusted AIX users: This command can perform privileged operations. Only privileged users can run privileged operations. For more information about authorizations and privileges, see Privileged Command Database in *Security*. For a list of privileges and the authorizations associated with this command, see the **/etc/security/privcmds** database file.

Examples

1. To print the file `memo` on the default printer, enter:
   ```
   enq memo
   ```
2. To print the file `prog.c` with page numbers, enter:
   ```
   pr prog.c | enq
   ```

   The **pr** command puts a heading at the top of each page that includes the date the file was last modified, the name of the file, and the page number. The **enq** command then prints the file.

3. To print a file with page numbers, reading from standard input, enter:

   ```
   pr x | enq  -P bill  -n  -r fn1 - fn3
   ```

   The dash (**-**) special file name tells the **enq** command to read from standard input. Normally the **enq** command will not read from standard input if there are file names on the command line. It also indicates the order in which to print things. The **pr** command creates a page numbered version of the file `x` and passes it to the **enq** command, which creates a temporary file containing that output in the **/var/spool/qdaemon** file.

   The **enq** command creates a job with four files and submits it to the queue named `bill`. It will print the `fn1` file twice. Then it will print whatever the output of the **pr** command was. Lastly it will print the file `fn3`. The four files are treated as one job for the purposes of burst pages. Notification is sent (the `-n` flag) when the job is complete. Since the `-r` flag was specified, the `fn1` and `fn3` files are removed at job completion. The temporary file created by the dash (**-**) file is always deleted.

   The **pr** command puts a heading at the top of each page that includes the date the file was last modified, the name of the file, and the page number. The **enq** command then prints the file.

4. To print the file `report` on the next available printer configured for the `fred` queue, enter:

   ```
   enq -P fred report
   ```

5. To print several files beginning with the prefix `sam` on the next available printer configured for the `fred` queue, enter:

   ```
   enq -P fred sam*
   ```

   All files beginning with the prefix `sam` are included in one print job. Normal status commands show only the title of the print job, which in this case is the name of the first file in the queue unless a different value was specified with the **-T** flag. To list the names of all the files in the print job, use the long status command **enq -A -L**.

6. To check the print queue to see if a file is still waiting to be printed, enter:

   ```
   enq  -q
   ```

   This command displays the status of the user's default queue. If the file is not yet printed, then it appears in the queue status listing. The system default queue is defined as the first queue in the **/etc/qconfig[.bin]** file. Users can have their own default override by setting and exporting the **PRINTER** environment variable.

7. To display the status of a nondefault queue, `lp0`, enter:

   ```
   enq  -q  -P lp0
   ```

8. To obtain the long queue status, enter:

   ```
   enq  -L
   ```

9. To obtain status on all queues, enter:

   ```
   enq  -A
   ```

10. To obtain long status on all queues, enter:

    ```
    enq  -A  -L
    ```

11. To obtain the status of the default queue, in wide format, enter:

    ```
    enq  -W
    ```

12. To obtain the wide status of all queues, enter:

    ```
    enq  -W  -A
    ```

13. To stop printing a job (a job is one or more files), enter:

```
enq  -x 413
```

This command cancels the request you made earlier to print a job. The number was obtained from the listing obtained by entering the **enq -q** command. If the job is currently being printed, the printer stops immediately. If the job has not been printed yet, it is removed from the queue so that it will not be printed. If the job is not in the queue, the **enq** command displays a message similar to the following:

```
no such request from you -- perhaps it's done?
```

14. To disconnect a printer from the queuing system, enter:

```
enq  -P lp0:dlp0  -D
```

Entering this command stops the **enq** command requests from being sent to the printer that serves the `lp0` queue. If a file is currently printing, it is allowed to finish. You must be able to execute the **qadm** command to run this command.

> **Note:** The printers serving a given queue are named by the device stanza name as it appears in the **/etc/qconfig[.bin]** file.

15. To print a file with page numbers using the **piobe** command backend on the default printer, enter:

```
enq  -o -p filename
```

The `-p` flag is not looked at by the **enq** command. The `-o` flag tells the **enq** command to pass the next item, which can be in quotes, to the backend unchanged. So, the **enq** command passes the `-p` flag to the **qdaemon** process, which in turn passes it to the backend **piobe**. The `-p` flag causes **piobe** to execute the **/usr/bin/pr** filter to apply page numbers to the document before giving data to the device. Multiple options can be given in quotes preceded by one `-o` flag or without quotes and individually preceded by more than one `-o` flag.

16. Assuming a **qconfig** file with the following information:

```
qname:
               device = fred
fred:
               file = /tmp/hello
               backend = /usr/bin/sh /usr/bin/diff
```

And given the following commands:

```
rm /tmp/hello
touch /tmp/hello
pr /etc/hosts|enq  -P qname:fred - /etc/hosts
```

The **qdaemon** process executes the `/usr/bin/diff` program with two arguments, one of which is a temporary file name and the other being the `/etc/hosts` file. The only difference between the two files is that one was run through the **pr** command. The `/tmp/hello` file will contain the differences between the two files. The **qdaemon** process does not create the `/tmp/hello` file if it does not exist.

17. The following command:

```
enq  -m'i want pink paper for this job' /etc/passwd
```

sends the specified operator message to the operator's console just before the print job is to print. The operator must respond to this message to continue or cancel the job.

```
enq  -M pink /etc/passwd
```

This command accomplishes the same thing, only the message is contained in a file called `pink`.

18. To cancel all jobs in the `fred` queue, enter:

    ```
    enq  -X  -P fred
    ```

    If the user who entered this command has root user authority, all the jobs from the `fred` queue are deleted. If the user does not have root user authority, only the users jobs are deleted from that queue.
19. To queue the file named `MyFile` and return the `MyFile` job number to the **jdf** file, enter:

    ```
    enq  -j MyFile
    ```
20. To hold print job number `310`, enter:

    ```
    enq  -h  -#310
    ```

    To release the hold on print job number `310`, enter:

    ```
    enq  -p -#310
    ```
21. To hold all the print jobs on queue `lp0`, enter:

    ```
    enq  -h  -P lp0
    ```

    To release the `lp0` queue, enter:

    ```
    enq  -p -P lp0
    ```
22. To hold all print jobs created by `fred`, enter:

    ```
    enq  -h  -u fred
    ```

    To release the print jobs created by `fred`, enter:

    ```
    enq  -p -u fred
    ```
23. To move job number `318` to queue `lp0`, enter:

    ```
    enq  -Q lp0  -#318
    ```

    The flags that control moving print jobs work in the same way as the flags that hold the print files. The hold flags and variables are illustrated in the preceding examples.

Files

/usr/sbin/qdaemon	Queuing daemon.
/etc/qconfig	Queue configuration file.
/var/spool/lpd/qdir/*	Queue requests.
/var/spool/lpd/stat/*	Information on the status of the devices.
/var/spool/qdaemon/*	Temporary copies of enqueued files.
/etc/qconfig.bin	Digested, binary version of the **/etc/qconfig** file.

Related Information

The **chquedev** command, **lsque** command, **mkque** command, **rmque** command.

The **qconfig** file.

Changing or showing queue characteristics in *Printers and printing*.

Printing administration in *Printers and printing*.

Printer-specific information in *Printers and printing*.

Installing support for additional printers in *Printers and printing*.

Print spooler in *Printers and printing*.

Virtual printer definitions and attributes in *Printers and printing*.

Printer colon file conventions in *Printers and printing*.

enroll Command

Purpose

Sets up a password used to implement a secure communication channel.

Syntax

enroll

Description

The **enroll** command establishes a password and secures a communication channel in which messages can only be read by the intended recipient. The password is used to receive secret mail.

The **enroll** command is used with the **xsend** and **xget** commands to send and receive secret mail. The **xsend** command sends secret mail. The **xget** command asks for your password and gives you your secret mail.

Examples

To set up a password, enter:

```
enroll
```

When prompted, enter your password. This allows other users on your system to send you secret mail. Use the **xget** command to read the secret mail.

Files

/var/spool/secretmail/User.key	Contains the encrypted key for the user.
/usr/bin/enroll	Contains the **enroll** command.

Related Information

The **mail** command, **xget** command, **xsend** command.

Mail in *Networks and communication management*.

Sending and receiving secret mail in *Networks and communication management*.

enscript Command

Purpose

Converts text files to PostScript format for printing.

Syntax

enscript [**-1 -2 -c -g -k -l -m -o -q -r -B -G -K -R**] [**-b** *Header*] [**-f** *Font*] [**-f0** *CodeSet*:*Font*] [**-f1** *CodeSet*:*Font*] [**-p** *Out*] [**-F** *Hfont*] [**-F0** *CodeSet*:*Font*] [**-F1** *CodeSet*:*Font*] [**-L** *Lines*] [**-M** *MediaName*] [**-X** *CodesetName*] [*SpoolerOptions*] [*File* ...]

Description

The **enscript** command reads a text file, converts it to PostScript format, and spools the file for printing on a PostScript printer. You can use this command to specify fonts, headings, limited formatting options, and spooling options.

For example:

```
enscript -daleph bubble.txt
```

prints a copy of the **bubble.txt** file on the printer called aleph, and

```
enscript -2r finder.c
```

prints a two-up landscape listing of the **finder.c** file on the default printer.

The **ENSCRIPT** environment variable can be used to specify defaults. The value of **ENSCRIPT** is parsed as a string of arguments before the arguments that are displayed on the command line. For example:

```
ENSCRIPT='-fTimes-Roman8'
```

sets your default body type size and font to 8-point Times Roman.

Information containing various media sizes for the **psdit** command and the **enscript** command are contained in the file **/usr/lib/ps/MediaSizes**.

The information required for each entry in the **MediaSizes** file can be obtained from the **PostScript Printer Description**, or **PPD**, file that matches the PostScript printer used with TranScript. The **PPD** files are available from Adobe Systems, Incorporated. The measurements extracted from the **PPD** files are expressed in a printer's measure called points. A printer's point is 1/72 of an inch.

Any line in the **MediaSizes** file beginning with an ASCII ***** (asterisk) is ignored when matching media-size names provided on the command line to the **enscript** command and the **psdit** command.

Each entry in the **MediaSizes** file contains either 8 or 9 fields. The first 8 fields are required for all entries. The 9th field is optional. Fields are separated by white space. The fields for each entry are as follows:

Field Name	Description
`EntryName`	Contains a character string to match against a media name provided with the **-M** flag with the **enscript** command or the **psdit** command.
`MediaWidth`	Specifies the media width in points.
`MediaDepth`	Specifies the media depth in points.
`ImageableLLX`	Specifies the imageable lower left-hand corner x coordinate in points.
`ImageableLLY`	Specifies the imageable lower left-hand corner y coordinate in points.
`ImageableURX`	Specifies the imageable upper right-hand corner x coordinate in points.
`ImageableURY`	Specifies the imageable upper right-hand corner y coordinate in points.
`PageRegionName`	Specifies the PostScript sequence for the particular printer to identify the size of the imageable area.
`PaperTrayName`	Specifies the PostScript sequence for the particular printer to select a particular paper/media tray. This field is optional. **Note:** The sequence can be multiple PostScript operators or words for both the `PageRegionName` field and the `PaperTrayName` field. To specify such a sequence, use the ASCII ″ (double quotation character) to delimit the entire sequence.

The following table shows examples of field entries in the **MediaSizes** file:

Name	Field Values
Letter	**Width** 612 **Depth** 792 **llx** 18 **lly** 17 **urx** 597 **ury** 776 **Page- Region- Name** Letter **Paper- Tray- Name** Letter
Legal	**Width** 612 **Depth** 1008 **llx** 18 **lly** 17 **urx** 597 **ury** 992 **Page- Region- Name** Legal **Paper- Tray- Name** Legal

PostScript Font Information

The PostScript Fonts for Transcript table shows the fonts available for the enscript command. The Font Name is specified with the **-F** and **-f encscipt** command flags. The alphabetic characters are case-sensitive:

PostScript Fonts for Transcript

Font Name	Font Family
AvantGarde-Book	AvantGarde
AvantGarde-Demi	AvantGarde
AvantGarde-DemiOblique	AvantGarde
AvantGarde-BookOblique	AvantGarde
Bookman-Demi	Bookman
Bookman-DemiItalic	Bookman
Bookman-Light	Bookman
Bookman-LightItalic	Bookman
Courier	Courier
Courier-Bold	Courier
Courier-BoldOblique	Courier
Courier-Oblique	Courier
Garamond-Bold	Garamond

PostScript Fonts for Transcript

Font Name	Font Family
Garamond-BoldItalic	Garamond
Garamond-Light	Garamond
Garamond-LightItalic	Garamond
Helvetica	Helvetica
Helvetica-Bold	Helvetica
Helvetica-Oblique	Helvetica
Helvetica-BoldOblique	Helvetica
Helvetica-Narrow	Helvetica
Helvetica-Narrow-Bold	Helvetica
Helvetica-Narrow-BoldOblique	Helvetica
Helvetica-Narrow-Oblique	Helvetica
LubalinGraph-Book	Lubalin
LubalinGraph-BookOblique	Lubalin
LubalinGraph-Demi	Lubalin
LubalinGraph-DemiOblique	Lubalin

Font Name	Font Family
Miryam-Iso	Miryam Iso
Miryam-IsoBold	Miryam Iso
Miryam-IsoBoldItalic	Miryam Iso
Miryam-IsoItalic	Miryam Iso
NarkissimIso	Narkissim Iso
NarkissimIso-Bold	Narkissim Iso
NarkissimIso-BoldItalic	Narkissim Iso
NarkissimIso-Italic	Narkissim Iso
NarkissTamIso	Narkiss Tam Iso
NarkissTamIso-Bold	Narkiss Tam Iso
NarkissTamIso-BoldItalic	Narkiss Tam Iso
NarkissTamIso-Italic	Narkiss Tam Iso
NewCenturySchlbk	NewCentury
NewCenturySchlbk-Bold	NewCentury
NewCenturySchlbk-Italic	NewCentury
NewCenturySchlbk-Roman	NewCentury
Optima	Optima
Optima-Bold	Optima
Optima-BoldOblique	Optima
Optima-Oblique	Optima
Palatino-Bold	Palatino
Palatino-BoldItalic	Palatino
Palatino-Italic	Palatino

Font Name	Font Family
Palatino-Roman	Palatino
Rokaa	Rokaa
Rokaa-Bold	Rokaa
Rokaa-BoldItalic	Rokaa
Rokaa-Italic	Rokaa

Font Name	Font Family
Setting	Setting
Setting-Bold	Setting
Setting-BoldItalic	Setting
Setting-Italic	Setting
ShalomIso	ShalomIso Iso
ShalomIso-Bold	ShalomIso Iso
ShalomIso-BoldItalic	ShalomIso Iso
ShalomIso-Italic	ShalomIso Iso
Souvenir-Demi	Souvenir
Souvenir-DemiItalic	Souvenir
Souvenir-Light	Souvenir
Souvenir-LightItalic	Souvenir
Times-Bold	Times
Times-BoldItalic	Times
Times-Italic	Times
Times-Roman	Times
Typing	Typing
Typing-Bold	Typing
Typing-BoldItalic	Typing
Typing-Italic	Typing
Symbol	(none)
ZapfChancery-MediumItalic	Zapf
ZapfDingbats	(none)

Parameters

SpoolerOptions Provides options for spooling the print file. The following are the *SpoolerOptions* flags:

{-d | -P}*Queue*
Queues the output to the named queue.

-n*Number*
Produces the specified number of copies. The default is 1.

-t*Title* Sets job title for use on the first banner page.

File Specifies the text file to be converted into PostScript format. If you leave this parameter blank, the **enscript** command reads from standard input.

Flags

-1 Sets in 1 column (the default).

-2 Sets in 2 columns.

-c Truncates (cuts) lines that are longer than the page width. Normally, long lines are wrapped around to the following line on the page.

-g Performs no function, but the **-g** flag is still accepted for backwards compatibility.

-k Enables page prefeed (if the printer supports it). This allows simple documents (such as program listings in a single font) to print somewhat faster by keeping the printer running between pages.

-l Simulates a line printer printing pages 66 lines long and omitting headers.

-m Sends mail after the files are printed.

-o Lists the missing characters if the **enscript** command cannot find characters in a font.

-q Causes the **enscript** command to not report about what it is doing. The **enscript** command cannot report on pages, destination, omitted characters, and so on. Fatal errors are still reported to the standard error output.

-r Rotates the output 90 degrees (landscape mode). Use this flag for output that requires a wide page or for program listings when used in conjunction with the **-2** flag. The following example shows one way to get program listings:

```
enscript -2r File . . .
```

-B Omits page headings.

-G Prints in gaudy mode, causing page headings, dates, and page numbers to be printed in a flashy style, at some slight performance expense.

-K Disables page prefeed (the default).

-R Prints in portrait mode (unrotated), which is the default.

-b*Header* Sets the string to be used for page headings to the value of the *Header* variable. The default header is constructed from the file name, its last modification date, and a page number.

-f*Font* Sets the font to be used for the body of each page. The default is Courier10, unless the two-column rotated mode is used, in which case it defaults to Courier7.

Notes:

1. A PostScript font name (such as Times-Roman, Times-BoldItalic, Helvetica, Courier).
2. A point size (1 point = 1/72 inch). Fonts are specified in this fashion: Courier-Bold8 is 8-point Courier Bold; Helvetica12 is 12-point Helvetica.

-f0 *Codeset*:*Font* Sets the character codeset name, which is written into the PostScript file, and the SBCS font to use for the body of each page. The default is determined by the **/usr/lib/ps/transcript.conf** configuration file for each locale.

-f1 *Codeset*:*Font* Sets the character codeset name, which is written into the PostScript file, and the MBCS font to use for the body of each page. The default is determined by the **/usr/lib/ps/transcript.conf** configuration file for each locale.

-p*Out* Causes the PostScript file to be written to the named file rather than being spooled for printing. As a special case, entering the following will send the PostScript file to standard output:

```
-p -
```

-F*Hfont* Sets the font to be used for page headings. The default is Courier Bold10.
Note: Font specifications have two parts:

- A PostScript font name (such as Times-Roman, Times-BoldItalic, Helvetica, Courier).
- A point size (1 point = 1/72 inch). Fonts are specified in this fashion: Courier-Bold8 is 8-point Courier Bold; Helvetica12 is 12-point Helvetica.

-F0 *Codeset*:*Font* Sets the character codeset name, which is written into the PostScript file, and the SBCS font to use for the header of each page. The default is determined by the **/usr/lib/ps/transcript.conf** configuration file for each locale.

-F1 *Codeset*:*Font*	Sets the character codeset name, which is written into the PostScript file, and the MBCS font to use for the header of each page. The default is determined by the **/usr/lib/ps/transcript.conf** configuration file for each locale.
-L*Lines*	Sets the maximum number of lines to print on a page. The **enscript** command usually computes how many lines to put on a page based on point size. (It might put fewer per page than requested by the **-L** flag.)
-M*MediaName*	Specifies a media name to use to determine the amount of imageable area on the paper. The name provided is matched against entries in the **MediaSizes** file. For instance, `-M legal` would request a legal size of paper as the imageable area. If this flag is not used, the default size is letter size, which is 8.5 inches wide by 11.0 inches deep (21.6 cent. wide by 27.9 cent. deep).
-X*CodesetName*	Specifies the code set for the input data. By default, the input code set is determined by the **nl_langinfo** subroutine. If this flag is used, the codeset is determined by the *CodesetName*.

International Character Support

All characters not found in a font will be replaced with the character ? (question mark). For a complete list of characters that were not found, use the **-o** flag. The **NLSvec** file provides information about character translation.

Environment Variables

ENSCRIPT	Specifies a string of options to be used by the **enscript** command.
LPDEST	Specifies a printer destination. The **-d** spooler option overrides this environment variable.
PSLIBDIR	Provides a path name of a directory to use instead of the **/usr/lib/ps** directory for the **enscript** command prologue and font metric files.
PSTEMPDIR	Provides a path name of temporary directory to use instead of the **/var/tmp** directory of spooled temporary files.
TRANSCRIPT	Provides the absolute path name of a file to use, instead of the **/usr/lib/ps/transcript.conf** configuration file, for MBCS handling.

Files

/usr/lib/ps/*.afm	Contains Adobe Font Metrics (AFM) files.
/usr/lib/ps/font.map	Contains the list of font names with their abbreviations.
/usr/lib/ps/enscript.pro	Contains prologue for **enscript** command files.
/usr/lib/ps/MediaSizes	Contains the default file used for media sizes.

Related Information

The **col** command, **eqn** command, **lp** command, **managefonts** command, **nroff** command, **pic** command, **pr** command, **ps630** command, **psdit** command, **refer** command, **tbl** command, **troff** command.

The **nl_langinfo** subroutine.

NLSvec File provides information about character translation.

---

entstat Command

Purpose

Shows ethernet device driver and device statistics.

Syntax

entstat [**-d -r -t**] *Device_Name*

Description

The **entstat** command displays the statistics gathered by the specified Ethernet device driver. The user can optionally specify that the device-specific statistics be displayed in addition to the device generic statistics. If no flags are specified, only the device generic statistics are displayed.

This command is also invoked when the **netstat** command is run with the **-v** flag. The **netstat** command does not issue any **entstat** command flags.

If an invalid *Device_Name* is specified, the **entstat** command produces an error message stating that it could not connect to the device.

Flags

-d Displays all the statistics, including the device-specific statistics.
-r Resets all the statistics back to their initial values. This flag can only be issued by privileged users.
-t Toggles debug trace in some device drivers.

Parameters

Device_Name The name of the Ethernet device, for example, **ent0**.

Statistic Fields

Note: Some adapters may not support a specific statistic. The value of non-supported statistic fields is always 0.

The statistic fields displayed in the output of the **entstat** command and their descriptions are:

Title Fields

`Device Type` Displays the description of the adapter type.
`Hardware Address` Displays the Ethernet network address currently used by the device.
`Elapsed Time` Displays the real time period which has elapsed since last time the statistics were reset. Part of the statistics may be reset by the device driver during error recovery when a hardware error is detected. There will be another Elapsed Time displayed in the middle of the output when this situation has occurred in order to reflect the time differences between the statistics.

Transmit Statistics Fields

`Packets` The number of packets transmitted successfully by the device.
`Bytes` The number of bytes transmitted successfully by the device.
`Interrupts` The number of transmit interrupts received by the driver from the adapter.
`Transmit Errors` The number of output errors encountered on this device. This is a counter for unsuccessful transmissions due to hardware/network errors.
`Packets Dropped` The number of packets accepted by the device driver for transmission which were not (for any reason) given to the device.
`Max Packets on S/W Transmit Queue` The maximum number of outgoing packets ever queued to the software transmit queue.

S/W Transmit Queue Overflow	The number of outgoing packets which have overflowed the software transmit queue.
Current S/W+H/W Transmit Queue Length	The number of pending outgoing packets on either the software transmit queue or the hardware transmit queue.
Broadcast Packets	The number of broadcast packets transmitted without any error.
Multicast Packets	The number of multicast packets transmitted without any error.
No Carrier Sense	The number of unsuccessful transmissions due to the no carrier sense error.
DMA Underrun	The number of unsuccessful transmissions due to the DMA underrun error.
Lost CTS Errors	The number of unsuccessful transmissions due to the loss of the Clear-to-Send signal error.
Max Collision Errors	The number of unsuccessful transmissions due to too many collisions. The number of collisions encountered exceeded the number of retries on the adapter.
Late Collision Errors	The number of unsuccessful transmissions due to the late collision error.
Deferred	The number of outgoing packets deferred during transmission. Deferred means that the adapter had to defer while trying to transmit a frame. This condition occurs if the network is busy when the adapter is ready to transmit. The adapter will only defer the first attempt to send a packet. After that the adapter will transmit the packet without checking. If the network is still busy then a collision will be recorded.
SQE Test	Contains the number of ″Signal Quality Error″ Tests (i.e. Heartbeat) performed successfully during transmission.
Timeout Errors	The number of unsuccessful transmissions due to adapter reported timeout errors.
Single Collision Count	The number of outgoing packets with single (only one) collision encountered during transmission.
Multiple Collision Count	The number of outgoing packets with multiple (2 - 15) collisions encountered during transmission.
Current HW Transmit Queue Length	The number of outgoing packets which currently exist on the hardware transmit queue.
CRC Errors	The number of incoming packets with the Checksum (FCS) error.
DMA Overrun	The number of incoming packets with the DMA overrun error.
Alignment Errors	The number of incoming packets with the alignment error.
No Resource Errors	The number of incoming packets dropped by the hardware due to the no resource error. This error usually occurs because the receive buffers on the adapter were exhausted. Some adapters may have the size of the receive buffers as a configurable parameter. Check the device configuration attributes (or smit helps) for possible tuning information.
Receive Collision Errors	The number of incoming packets with the collision errors during the reception.
Packet Too Short Errors	The number of incoming packets with the length error indicating that the packet size is less than the Ethernet minimum packet size.
Packet Too Long Errors	The number of incoming packets with the length error indicating that the packet size is bigger than the Ethernet maximum packet size.
Packets Discarded by Adapter	The number of incoming packets dropped by the hardware for any other reasons.
Receiver Start Count	The number of times that the receiver (receive unit) on the adapter has been started.

Receive Statistics Fields

Packets	The number of packets received successfully by the device.

Bytes	The number of bytes received successfully by the device.
Interrupts	The number of receive interrupts received by the driver from the adapter.
Receive Errors	The number of input errors encountered on this device. This is a counter for unsuccessful reception due to hardware/network errors.
Packets Dropped	The number of packets received by the device driver from this device which were not (for any reason) given to a network demuxer.
Bad Packets	The number of bad packets received (i.e. saved) by the device driver.
Broadcast Packets	The number of broadcast packets received without any error.
Multicast Packets	The number of multicast packets received without any error.
CRC Errors	The number of incoming packets with the Checksum (FCS) error.
DMA Overrun	The number of incoming packets with the DMA overrun error.
Alignment Errors	The number of incoming packets with the alignment error.
No Resource Errors	The number of incoming packets dropped by the hardware due to the no resource error.
Receive Collision Errors	The number of incoming packets with the collision errors during the reception.
Packet Too Short Errors	The number of incoming packets with the length error indicating that the packet size is less than the Ethernet minimum packet size.
Packet Too Long Errors	The number of incoming packets with the length error indicating that the packet size is bigger than the Ethernet maximum packet size.
Packets Discarded by Adapter	The number of incoming packets dropped by the hardware for any other reasons.
Receiver Start Count	The number of times that the receiver (receive unit) on the adapter has been started.

General Statistics Fields

No mbuf Errors	The number of times that mbufs were not available to the device driver. This usually occurs during receive operations when the driver must obtain mbuf buffers to process inbound packets. If the mbuf pool for the requested size is empty, the packet will be discarded. The **netstat -m** command can be used to confirm this.
Adapter Reset Count	The number of times that the adapter has been restarted (re-initialized).
Driver Flags	The device driver internal status flags that are currently turned on.

Device Specific Statistics Fields

This part of the display may be different for each type of the adapter. It may contain adapter specific information and some extended statistics that were not included in the generic statistics. Some adapters may not have any device specific statistics.

Security

Attention RBAC users and Trusted AIX users: This command can perform privileged operations. Only privileged users can run privileged operations. For more information about authorizations and privileges, see Privileged Command Database in *Security*. For a list of privileges and the authorizations associated with this command, see the **/etc/security/privcmds** database file.

Examples

1. To display the device generic statistics for **ent0**, enter:

   ```
   entstat ent0
   ```

   This produces the following output:

   ```
   ETHERNET STATISTICS (ent0) :
   Device Type: Ethernet High Performance LAN Adapter
   Hardware Address: 02:60:8c:2e:d0:1d
   Elapsed Time: 0 days 0 hours 8 minutes 41 seconds

   Transmit Statistics:     Receive Statistics:
   --------------------     -------------------
   Packets: 3               Packets: 2
   Bytes: 272               Bytes: 146
   Interrupts: 3            Interrupts: 2
   Transmit Errors: 0       Receive Errors: 0
   Packets Dropped: 0       Packets Dropped: 0
   Max Packets on S/W       Bad Packets: 0
   Transmit Queue:0
   S/W Transmit Queue
   Overflow: 0
   Current S/W+H/W Transmit
   Queue Length: 0

   Broadcast Packets: 2     CRC Errors: 0
   Multicast Packets: 0     Broadcast Packets: 1
   No Carrier Sense: 0      Multicast Packets: 0
   DMA Underrun: 0          DMA Overrun: 0
   Lost CTS Errors: 0       Alignment Errors: 0
   Max Collision Errors: 0  No Resource Errors: 0
   Late Collision Errors: 0 Receive Collision Errors: 0
   Deferred: 0              Packet Too Short Errors: 0
   SQE Test: 0              Packet Too Long Errors: 0
   Timeout Errors: 0        Packets Discarded by Adapter: 0
   Single Collision         Receiver Start Count: 1
   Count: 0
   Multiple Collision Count: 0
   Current HW Transmit Queue
   Length: 0

   General Statistics:
   -------------------
   No mbuf Errors: 0
   Adapter Reset Count: 0
   Driver Flags: Up Broadcast Running Simplex
   ```

2. To display the Ethernet device generic statistics and the ethernet device-specific statistics for **ent0**, enter:

   ```
   entstat -d ent0
   ```

   This produces the following output:

   ```
   ETHERNET STATISTICS (ent0) :
   Device Type: Ethernet High Performance LAN Adapter
   Hardware Address: 02:60:8c:2e:d0:1d
   Elapsed Time: 0 days 2 hours 6 minutes 30 seconds

   Transmit Statistics:      Receive Statistics:
   --------------------      -------------------
   Packets: 3                Packets: 2
   Bytes: 272                Bytes: 146
   Interrupts: 3             Interrupts: 2
   Transmit Errors: 0        Receive Errors: 0
   Packets Dropped: 0        Packets Dropped: 0
   Max Packets on S/W        Receiver Start Count: 1
   ```

```
Transmit Queue:0
Bad Packets: 0
S/W Transmit Queue Overflow: 0
Current S/W+H/W Transmit Queue Length: 0

Broadcast Packets: 0      Broadcast Packets: 0
Multicast Packets: 0      Multicast Packets: 0
No Carrier Sense: 0       CRC Errors: 0
DMA Underrun: 0           DMA Overrun: 0
Lost CTS Errors: 0        Alignment Errors: 0
Max Collision Errors: 0   No Resource Errors: 0
Late Collision Errors: 0  Receive Collision Errors: 0
Deferred: 0               Packet Too Short Errors: 0
SQE Test: 0               Packet Too Long Errors: 0
Timeout Errors: 0         Packets Discarded by Adapter: 0
Single Collision Count: 0 Receiver Start Count: 1
Multiple Collision Count: 0
Current HW Transmit Queue Length: 0

General Statistics:
-------------------
No mbuf Errors: 0
Adapter Reset Count: 0
Driver Flags: Up Broadcast Running Simplex

Ethernet High Performance LAN Adapter Specific Statistics:
----------------------------------------------------------
Receive Buffer Pool Size: 37
Transmit Buffer Pool Size: 39
In Promiscuous Mode for IP Multicast: No
Packets Uploaded from Adapter: 0
Host End-of-List Encountered: 0
82586 End-of-List Encountered: 0
Receive DMA Timeouts: 0
Adapter Internal Data: 0x0 0x0 0x0 0x0 0x0
```

Related Information

The **atmstat** command, **fddistat** command, **netstat** command, **tokstat** command.

---

env Command

Purpose

Displays the current environment or sets the environment for the execution of a command.

Syntax

To Display Multiple Environment Variables

env [**-i** | **-**] [*Name*=*Value*]... [*Command* [*Argument* ...]]

To Display A Single Environment Variable

env [*Name*]

Description

The **env** command allows you to display your current environment or run a specified command in a changed environment.

If no flags or parameters are specified, the **env** command displays your current environment, showing one *Name*=*Value* pair per line.

Flags

-i Ignores the inherited environment and invokes the command specified by the *Command* parameter with the environment specified by the *Name*=*Value* parameters.

Parameters

Name=Value You can run a command in a modified version of the current environment by specifying one or more *Name=Value* parameters. Use the **-i** flag if you wish to replace the entire current environment with the specified *Name =Value* parameters. In either case, environment changes are effective only while the specified command is running.

Command The *Command* parameter has an optional *Argument* variable. If the specified command is one of the Korn shell special built-in commands, results are unspecified. Korn shell built-in commands are described in the **ksh** command.

Exit Status

If the *Command* parameter is specified, the exit status of the **env** command is the exit status of the command specified in the *Command* parameter. Otherwise, the **env** command exits with one of the following values:

0 The **env** command completed successfully.
1-125 An error occurred in the **env** command.
126 The command specified by the *Command* parameter was found, but could not be invoked.
127 The command specified by the *Command* parameter was not found.

Examples

1. To change the **TZ** environment variable while running the **date** command, type:

   ```
   TZ=MST7MDT date
   ```

   OR

   ```
   env TZ=MST7MDT date
   ```

   Each of these commands displays the time in mountain time and the current date. The two commands shown are equivalent. When the **date** command is finished, the previous value of the **TZ** environment variable takes effect again.
2. To run the **make** command in an environment that consists only of definitions for the **PATH, IDIR,** and **LIBDIR** environment variables, type:

   ```
   env -i PATH=$PATH IDIR=/$HOME/include LIBDIR=/$HOME/lib make
   ```

   You must specify the **PATH** environment variable so that the shell can find the **make** command. When the **make** command is finished, the previous environment takes effect.

Files

/usr/bin/env Contains the **env** command.

Related Information

The **printenv** command, **ksh** command.

The **environment** file.

The **profile** file format.

The **exec** subroutines.

Commands in *Operating system and device management*.

Profiles overview in *Operating system and device management*.

Shells in *Operating system and device management*.

epkg Command

Purpose

Creates interim fix (interim fix) packages that can be installed by the interim fix manager, **emgr**.

Syntax

epkg [**-w** *WorkDirectory*] [**-p** *PrerequisiteFile*] [**-d** *DescriptionFile*] [**-e** *interim fixControlFile*] [**-g** *PrerequisiteFile*] [**-l** *LockFile*] [**-S** *SupersedeFile*] [**-u** {**y**|**n**}]] [**-r** {**y**|**n**|**o**}] [**-s**] [**-T** {**y**|**n**}] [**-X**] [**-v**] *interim fixLabel*

Description

The **epkg** tool can be run in two different modes: *interactive* and *template-based*. The interactive mode prompts you with several questions and constructs the interim fix package based on the answers. The template-based mode uses an interim fix control file that is provided with the answers to questions that are asked in interactive mode. The interim fix package is installed by the interim fix manager, which is started with the **emgr** command.

Interactive mode

The **epkg** command runs in interactive mode by default. The only required parameter is the interim fix label. If you interrupt an **epkg** session, the interim fix control file will be saved. If you start a new session with the same interim fix label, you will be asked whether you want to keep working with the previous interim fix control file. To provide this information before you start the interactive **epkg** session, run **epkg** with the **-u** flag.

The **epkg** command maintains a record of the question order and allows you to navigate between questions by using subcommands. Also, the **epkg** command remembers the previous answer you provided and sets that answer as the default answer. The **epkg** subcommands are described in the Subcommands section.

After you answer all the questions, the **epkg** command verifies the interim fix control file and creates a compressed tar package that can be installed with the **emgr** command.

Using the interim fix control file template

You can create interim fix packages noninteractively by using an interim fix control file as a template. The following is an example of a completed interim fix control file:

```
# interim fix control file complete example
ABSTRACT=This is a test of epkg.
PRE_INSTALL=/tmp/pre_install
POST_INSTALL=.
PRE_REMOVE=/tmp/pre_remove
POST_REMOVE=.
REBOOT=yes
PREREQ=.
DESCRIPTION=/tmp/description
EFIX_FILES=2
```

```
EFIX_FILE:
        EFIX_FILE_NUM=1
        SHIP_FILE=/home/test/ls
        TARGET_FILE=/usr/bin/ls
        TYPE= 1
        INSTALLER= 1
        ACL= DEFAULT
        AR_MEM=.

EFIX_FILE:
        EFIX_FILE_NUM=2
        SHIP_FILE=/home/test/mystrcat.o
        TARGET_FILE=/usr/ccs/lib/libc.a
        TYPE= 2
        INSTALLER= 1
        ACL= root:system:555
        AR_MEM=strcat.o
```

The interim fix control file values, are as follows:

ABSTRACT
Briefly describes the interim fix package. The abstract is limited to 38 bytes.

PRE_INSTALL
Specifies the location of a script that is run after the installation preview and before any interim fix files are installed. Failure in the PRE_INSTALL script will cause the interim fix package installation to be aborted. This component is optional.

POST_INSTALL
Specifies the location of a script that is run after all interim fix files have been successfully installed. This component is optional.

PRE_REMOVE
Specifies the location of a script that is run after the removal preview and before any interim fix files are removed during a remove operation. This component is optional.

POST_REMOVE
Specifies the location of a script that is run after interim fix files are removed during a remove operation. This component is optional.

REBOOT
Specifies whether a reboot is required for this interim fix. Allowable values are yes or no. If this value is set to yes, the **emgr** command will make changes as necessary to the boot image and issue a message instructing the user to reboot after installation.

PREREQ
Specifies the location of a file that contains **installp** prerequisites. This component is optional.

DESCRIPTION
Specifies the location of a file that contains a detailed description of the interim fix package that is being installed.

EFIX_FILES
Specifies the total number of files in the interim fix.

EFIX_FILE_NUM
Specifies the number of the file in the interim fix. Each file in the interim fix must have a unique number, from 1 to 200. The **epkg** command can support a maximum of 200 files per interim fix.

SHIP_FILE
Specifies the location of a file that **epkg** will archive into the interim fix package. You can specify either an absolute path or a relative path to this file.

TARGET_FILE
Specifies the location where the **SHIP_FILE** will be installed. This location is on the system where the interim fix package will be installed. You must specify an absolute path to this file. If this file is part of a registered package, such as an RPM Package Manager (RPM) or **installp** package, you must specify the tracked location.

TYPE Specifies the type of file that is being installed. The valid choices are as follows:

1 File (standard or executable)

2 Library or archive member

INSTALLER
Specifies the type of installer, if any, that will track the interim fix package. The valid choices are as follows:

1 Currently tracked by **installp**

2 Currently tracked by RPM

3 Currently tracked by **ISMP**

4 Currently tracked by another installer

5 This is a new file that will be tracked by **installp**

6 New file that will be tracked by RPM

7 New file that will be tracked by **ISMP**

8 New file that will be tracked by another installer

9 Not tracked by any installer

ACL Specifies the access attributes (mode and ownership) for the file. If this attribute is set to **DEFAULT**, the **emgr** command maintains the current permissions of the file to be replaced. However, if the target file is a new file or if the user wants to specify permissions with the **-v** flag, the **ACL** attribute can be entered with the syntax *Owner*:*Group*:*OctalModes*, similar to the following:

```
ACL= root:system:555
```

AR_MEM
Specifies the name of the archive member. This option is only valid if **TYPE**=2. In this case, **SHIP_FILE** is the local location of the archive member that is being shipped, **TARGET_FILE** is the target archive, and **ACL** applies to the archive member. For example, the following value settings would make the local file **myshr.o** the member **shr.o** in the target archive **/usr/ccs/lib/libc.a**:

```
TYPE=2
SHIP_FILE=/home/myshr.o
TARGET_FILE=/usr/ccs/lib/libc.a
AR_MEM=shr.o
```

BUILD_BOOTIMAGE
Specifies whether the boot image needs to be rebuilt. Allowable values are yes or no. A reboot is required if this field is set to yes. If this field is set to yes and the **REBOOT** field is set to no, **epkg** returns an error.

E2E_PREREQ
Specifies the location of the interim fix prerequisite file in the interim fix control file.

PKGLOCKS
Specifies the local file location of the package lock file in the interim fix control file.

SUPERSEDE
Specifies the local file location of the superseded file in the interim fix control file.

FIXTESTED
Specifies whether this interim fix has been tested. Allowable values are yes or no.

Support for interim fix Superseding

The packager can specify a file containing the interim fix label names that are to be superseded when an epkg is installed. This will cause the **emgr** command to remove any interim fix labels that are specified in this file (if they are installed) before installing the interim fix package. Failure to remove an installed superseded interim fix will abort the installation of the interim fix package. The maximum supported number of superseded labels is 32. The packager can specify the supersede file with the **epkg** command in the following ways:

- Specify the file location with the **-S** *supersede_file* flag. For example:

  ```
  epkg -S /tmp/superseded.epkg myefix
  ```

- The **epkg** command will prompt for the superseded file if the extended options flag (**-v**) is used in interactive mode. For example:

  ```
  Enter the location for the supersede file or "." to skip.
  -> /tmp/superseded.epkg
  ```

- Set the SUPERSEDE attribute to the local file location of the superseded file in the interim fix control file. For example:

  ```
  SUPERSEDE=/tmp/superseded.epkg
  ```

The format of the superseded file is one interim fix label to be superseded per line. Comments beginning with a # sign and leading white space are ignored. For example:

```
# Requisites for efix myefix3
myefix1
myefix2
```

Support for interim fix prereqs and xreqs

The packager can specify a file containing the interim fix label names of interim fixes that are requisites to the interim fix package being installed. This will cause the **emgr** command to check if the interim fix label is installed (PREREQ). If the requisite is not installed, the **emgr** command will abort installation of the interim fix package. The user can also specify an XREQ interim fix label. This will cause the **emgr** command *not* to install the interim fix if the named xreq interim fix is installed.

The packager can specify the interim fix prerequisite file with the **epkg** command in the following ways:

- Specify the file location with the **-g** *efix_prereq_file* flag. For example:

  ```
  epkg -g /tmp/efixprereq.epkg myefix
  ```

- The **epkg** command will prompt for the interim fix prereq file if the extended options flag (**-v**) is used in interactive mode. For example:

  ```
  Enter the location for the efix prerequisite file or "." to skip.
  -> /tmp/efixprereq.epkg
  ```

- Set the E2E_PREREQ attribute to the local file location of the interim fix prerequisite file in the interim fix control file. For example:

  ```
  E2E_PREREQ=/tmp/efixprereq.epkg
  ```

The format of the interim fix prerequisite file entries is as follows:

EfixLabel RequisiteType: PREREQ/XREQ

For example:

```
oldefix1 PREREQ  # Make sure oldefix1 is already installed

oldefix4 XREQ    # Make sure oldefix4 is NOT installed
```

The maximum number of supported interim fix prerequisites is 32.

interim fix Output and Topology

The **emgr -d** flag displays the contents and topology of the interim fix package. The **-d** option will work with the **-v** verbose option. Valid levels of verbosity are 1-3.

Verbosity level 1 (default) will display:

- LABEL
- EFIX FILES
- TARGET LOCATION

Verbosity level 2 will display:

- All level 1 output
- ABSTRACT
- REBOOT
- PRE-REQUISITES
- PRE_INSTALL
- POST_INSTALL
- PRE_REMOVE
- POST_REMOVE
- FILE TYPE

Verbosity level 3 will display:

- All level 2 output
- PACKAGING DATE
- VUID
- SIZE
- ACL
- CKSUM
- PACKAGE
- EFIX DESCRIPTION
- CONTENTS OF INSTALL SCRIPTS (if text files)

For example:

- To get level 1 verbosity output on interim fix package `test.102403.epkg.Z`, type:

  ```
  emgr -d test.102403.epkg.Z
  ```

- To get level 3 verbosity output on interim fix package `test.102403.epkg.Z`, type:

  ```
  emgr -v3 -d test.102403.epkg.Z
  ```

Support for Additional Package Locking

The packager can specify a file containing package names that should be locked by the **emgr** command in addition to those that are automatically locked based on file ownership. The packager must specify the name of the package, the installer, and the type of package lock action (ALWAYS/IFINST). The packager can specify the package lock file using the **epkg** command in the following ways:

- Specify the file location with the **-l** *pkg_locks_file* flag. For example:

  ```
  epkg -l /tmp/pkglock.epkg myefix
  ```

- The **epkg** command will prompt for the package locks file if the extended options flag (**-v**) is used. For example:

  ```
  Enter the location for the package locks file or "." to skip.
  -> /tmp/pkglock.epkg
  ```

- Set the PKGLOCKS attribute to the local file location of the package lock file in the interim fix control file. For example:

  ```
  PKGLOCKS=/tmp/pkglock.epkg
  ```

The format of the package locks file is as follows:

PackageName PackageAction PackageType

where *PackageName* is the name of the package to be locked and *PackageAction* is one of the following:

ALWAYS	Always attempt to lock this package. Failure to lock the package results in installation failure.
IFINST	Attempt to lock this package only if the package is installed. Failure to lock an *installed* package results in installation failure.

PackageType is installp (default), rpm, ISMP, other.

Note: Only installp locking is supported.

The maximum number of supported package lock entries is 32.

Example:

```
bos.rte.lvm ALWAYS installp
bos.games   IFINST installp
```

In the above example, the **emgr** command will always attempt to lock **bos.rte.lvm** during installation and will unlock it on removal. The **emgr** command will lock **bos.games** if, and only if, it is installed, and will unlock it on removal (if locked).

Support for the bosboot Option

The **epkg** command reboot options include rebooting without rebuilding the boot image.

The user can specify a reboot without bosboot in the following ways:

- The o argument for the **epkg -r** flag indicates that reboot (″only″) is required, but the **emgr** command should not call bosboot (that is, rebuild the boot image).
- The reboot prompt in interactive mode indicates the following choices:

  ```
  Select reboot policy for this efix package:
  1) Reboot is NOT required.
  2) Reboot is required. The boot image will be rebuilt.
  3) Reboot is required. The boot image will NOT be rebuilt.
  ```

- Set the BUILD_BOOTIMAGE and REBOOT attribute to ″yes″ or ″no″ in the interim fix control file. The following REBOOT and BUILD_BOOTIMAGE options are supported:

REBOOT=no & BUILD_BOOTIMAGE=no	Reboot is NOT required.
REBOOT=yes & BUILD_BOOTIMAGE=yes	Reboot is required. The boot image will be rebuilt.
REBOOT=yes & BUILD_BOOTIMAGE=no	Reboot is required. The boot image will *not* be rebuilt.

Note: `REBOOT=no & BUILD_BOOTIMAGE=yes` will result in an error from the **epkg** command.

Flags

-d *DescriptionFile*	Specifies the file containing the interim fix description.
-e *interim fixControlFile*	Specifies the interim fix control file that controls how the interim fix is constructed.

-g *PrerequisiteFile*	Specifies the location of the interim fix prerequisite file that contains the interim fix label names. These labels are required before an interim fix package is installed.
-l *LockFile*	Specifies the location of the locked file that contains the package names. These packages are locked by the **emgr** command or automatically based on file ownership.
-p *PrerequisiteFile*	Specifies the file containing installp prerequisites.
-r {y|n|o}	Sets the **epkg REBOOT** attribute. This causes the **emgr** command to make changes as necessary to the boot image and issue a message instructing the user to reboot after installation. The **y** argument specifies that a reboot and a **bosboot** are required. The **n** argument specifies that a reboot is not required. The **o** argument indicates that a reboot is required, but **emgr** should not call **bosboot**.
-S *SupersedeFile*	Specifies the location of the interim fix supersede file that contains the interim fix label names. These labels are to be superseded when an **epkg** is installed.
-s	Causes the **epkg** command to skip questions regarding scripts and the prerequisite file.
-T	Specifies whether this interim fix was tested. Allowable values are yes or no. The default is no.
-u {**yes**|**no**}	Specifies whether you will use an existing interim fix control file.
-v	Causes the **epkg** command to ask more questions for extended options. This includes asking you to specify permissions on all interim fix files.
-w *WorkDirectory*	Specifies the alternate work directory that the **epkg** command will use. The default work directory is **$HOME/epkgwork**.
-X	Causes the **emgr** command to automatically expand file systems when the interim fix is installed, if space is required and expansion is possible.

Parameters

interim fixLabel

Specifies a string that uniquely identifies this interim fix package. The maximum length of an interim fix label is 10 bytes.

Note: The interim fix manager requires each interim fix label on the system to be unique.

Subcommands

b! Returns to the previous question.

s! Shows the status of the current interim fix control file

q! Quits without saving the interim fix control file. (Using the Ctrl+C key sequence causes the **epkg** command to ask you whether you want to save the interim fix control file.)

h! Displays help information for the current question.

Exit Status

0 The **epkg** command operations completed successfully.

>0 An error occurred.

Examples

1. To run the **epkg** command in interactive mode and create an interim fix package with the interim fix label of **myfix**, type:

   ```
   epkg myfix
   ```

2. To create and interim fix package with the interim fix label of **myfix** using an existing interim fix control file named **/tmp/ecfile**, type:

   ```
   epkg -e /tmp/ecfile myfix
   ```

3. To create an interim fix package with the interim fix label of **myfix** and specify prerequisite file **/tmp/prereq**, description **/tmp/description**, and extended options, type:

```
epkg -v -p /tmp/prereq -d /tmp/description myfix
```

Files

/usr/sbin/epkg	Contains the **epkg** command.

Related Information

The **emgr** command.

Installing optional software products and service updates in *Installation and migration*.

---

eqn Command

Purpose

Formats mathematical text for the **troff** command.

Syntax

eqn [**-d** *Delimiter1Delimiter2*] [**-f** *Font*] [**-p** *Number*] [**-s** *Size*] [**-T** *Name*] [**—**] [*File* ... | **-**]

Description

The **eqn** command is a **troff** preprocessor for typesetting mathematical text on a phototypesetter or comparable device. The output of the **eqn** command is generally piped into the **troff** command, as follows:

eqn [*Flag*...] *File*... **|** **troff** [*Flag*...] **|** [*Typesetter*]

The **eqn** command reads files specified by the *File* parameter. It reads standard input when a - (minus sign) is specified as the last parameter. A line beginning with the **.EQ** macro marks the start of equation text. The end of equation text is marked by a line beginning with the **.EN** macro. These lines are not altered by the **troff** command, so they can be defined in macro packages to provide additional formatting function such as centering and numbering.

Keywords

The following are keywords known to both the **eqn** and **neqn** commands.

above	**dot**	**gsize**	**over**	**tdefine**
back	**dotdot**	**hat**	**pile**	**tilde**
bar	**down**	**italic**	**rcol**	**to**
bold	**dyad**	**lcol**	**right**	**under**
ceiling	**fat**	**left**	**roman**	**up**
ccol	**floor**	**lineup**	**rpile**	**vec**
col	**font**	**lpile**	**size**	
cpile	**from**	**mark**	**sqrt**	
define	**fwd**	**matrix**	**sub**	
delim	**gfont**	**ndefine**	**sup**	

Keywords recognized by the **eqn** command can be set apart with spaces, tabs, new lines, braces, double quotes, tildes, and circumflexes. Use { } (braces) for groupings; anywhere you can use a single character, such as X, you can substitute a complicated construction enclosed in braces. The ~ (tilde) represents a full space in the output, and the ^ (circumflex) represents a half-space.

Produce subscripts and superscripts using the **sub** and **sup** keywords. Produce fractions with the **over**keyword. Produce square roots with the **sqrt** keyword.

Introduce lower and upper limits using the **from** and **to** keywords. Produce delimiters (such as left and right brackets and braces) of the correct height using the **left** and **right** keywords. Legal characters after

the **left** and **right** keywords are braces, brackets, bars, **c** and **f** for ceiling and floor, and **" "** (double quotes) for nothing at all (which is useful for a right-side-only bracket). A **left** character does not need a matching **right** character, but a **right** character must have a matching **left** character.

Vertical lists (piles) of things are made with the **pile**, **lpile**, **cpile**, and **rpile** keywords. Piles can have arbitrary numbers of elements. The **lpile** keyword left-justifies, the **pile** and **cpile** keywords center (but with different vertical spacing), and the **rpile** keyword right-justifies. Matrices are made with the **matrix** keyword. In addition, there is an **rcol** keyword for a right-justified column.

Diacritical marks are made with the **dot**, **dotdot**, **hat**, **tilde**, **bar**, **vec**, **dyad**, and **under** keywords.

You can change point sizes and fonts with the **size** *Number* (or **size +/-***Number*), **roman**, **italic**, **bold**, and **font** *Number* keywords. You can change point sizes and fonts globally in a document with the **gsize** *Number* and **gfont** *Number* keywords, or with the command-line **-s***Number* and **-f***Number* flags.

Normally, subscripts and superscripts are reduced by three points from the previous size. You can change this with the command-line **-p***Number* flag.

You can line up successive display parameters. Place the **mark** keyword before the desired lineup point in the first equation; place the **lineup** keyword where it is to line up vertically in subsequent equations.

You can define shorthands or redefine existing keywords with the **define** keyword; for example:

```
define  Thing%Replacement%
```

The preceding example defines a new token called *Thing* that is replaced by *Replacement* whenever it appears thereafter. The % (percent sign) can be any character that does not occur in *Replacement*.

Keywords such as **sum**, **int**, **inf**, and shorthands such as **>=**, **!=**, and **->** are recognized. Greek letters are spelled out in the desired case, as in **alpha** or **GAMMA**. Mathematical words such as **sin**, **cos**, and **log** are made Roman automatically. The **troff** command 4-character escapes, such as **\(dd**, which produces the double dagger, can be used anywhere. Strings enclosed in **" "** (double quotes) are passed through untouched. This permits keywords to be entered as text, and can always be used to communicate with the **troff** command.

Flags

-d*Delimiter1Delimiter2*	Sets two ASCII characters, *Delimiter1* and *Delimiter2,* as delimiters of the text to be processed by the **eqn** command, in addition to the input enclosed by the **.EQ** and **.EN** macros. The text between these delimiters is treated as input to the **eqn** command. **Note:** Within a file, you can also set delimiters for **eqn** text using the **delim** *Delimiter1Delimiter2*command. They are turned off by the **delim off** command. All text not between **.EQ** and **.EN** macros is passed through unprocessed.
-f*Font*	Changes font in all the **eqn** command processed text to the value specified by the *Font* variable. The *Font* value (a font name or position) must be one or two ASCII characters.
-p*Number*	Reduces subscripts and superscripts the specified number of points in size (the default is 3).
-s*Size*	Changes point size in all the **eqn** command processed text to the value specified by the *Size* variable.
-T*Name*	Prepares the output for the specified printing device. Terminal Names for Phototypesetter or Comparable Devices provides *Name* variables. The default is **ibm3816**.
-	Forces input to be read from standard input.

—	(double dash) Indicates the end of flags.

Files

/usr/share/lib/pub/eqnchar	Contains special character definitions.

Related Information

The **checkeq** command, **mmt** command, **mvt** command, **neqn** command, **nroff** command, **tbl** command, **troff** command.

The **eqnchar** file format contains special character definitions for the **eqn** and **neqn** commands.

The **.EQ** and **.EN** macros, **mm** macro package, **mv** macro package.

---

errclear Command

Purpose

Deletes entries from the error log.

Syntax

errclear [**-d** *ErrorClassList*] [**-i** *File*] [**-J** *ErrorLabel* [*,Errorlabel*]] | [**-K** *ErrorLabel* [*,Errorlabel*]] [**-l** *SequenceNumber*] [**-m** *Machine*] [**-n** *Node*] [**-N** *ResourceNameList*] [**-R** *ResourceTypeList*] [**-S** *ResourceClassList*] [**-T** *ErrorTypeList*] [**-y** *FileName*] [**-j** *ErrorID* [**,***ErrorID*]] | [**-k** *ErrorID* [**,***ErrorID*]] *Days*

Description

The **errclear** command deletes error-log entries older than the number of days specified by the *Days* parameter. To delete all error-log entries, specify a value of **0** for the *Days* parameter.

If the **-i** flag is not used with the **errclear** command, the error log file cleared by **errclear** is the one specified in the error log configuration database. (To view the information in the error log configuration database, use the **errdemon** command.)

> **Note:** The **errclear** command clears the specified entries, but does not decrease the error log file size.

You can use the System application in Web-based System Manager (wsm) to change system characteristics. You could also use the System Management Interface Tool (SMIT) **smit errclear** fast path to run this command.

Flags

-d *List*	Deletes error-log entries in the error classes specified by the *List* variable. The *List* variable values can be separated by , (commas), or enclosed in " " (double quotation marks) and separated by , (commas) or space characters. The valid *List* variable values are **H** (hardware), **S** (software), **O** (**errlogger** messages), and **U** (undetermined).
-i *File*	Uses the error-log file specified by the *File* variable. If this flag is not specified, the **errclear** command uses the value from the error-log configuration database.

-j *ErrorID*[,*ErrorID*]	Deletes the error-log entries specified by the *ErrorID* (error identifier) variable. The *ErrorID* variable values can be separated by , (commas), or enclosed in ″ ″ (double quotation marks) and separated by , (commas) or space characters.
-J *ErrorLabel*	Deletes the error-log entries specified by the *ErrorLabel* variable. The *ErrorLabel* variable values can be separated by , (commas), or enclosed in ″ ″ (double quotation marks) and separated by , (commas) or space characters.
-k *ErrorID*[,*ErrorID*]	Deletes all error-log entries except those specified by the *ErrorID* (error identifier) variable. The *ErrorID* variable values can be separated by , (commas), or enclosed in ″ ″ (double quotation marks) and separated by , (commas) or space characters.
-K *ErrorLabel*	Deletes all error-log entries except those specified by the *ErrorLabel* variable. The *ErrorLabel* variable values can be separated by , (commas), or enclosed in ″ ″ (double quotation marks) and separated by , (commas) or space characters.
-l *SequenceNumber*	Deletes error-log entries with the specified sequence numbers. The *SequenceNumber* variable values can be separated by , (commas), or enclosed in ″ ″ (double quotation marks) and separated by , (commas) or space characters.
-m *Machine*	Deletes error-log entries for the machine specified by the *Machine* variable. The **uname -m** command returns the value of the *Machine* variable.
-n *Node*	Deletes error-log entries for the node specified by the *Node* variable. The **uname -n** command returns the value of the *Node* variable.
-N *List*	Deletes error-log entries for the resource names specified by the *List* variable. The *List* variable is list of names of resources that have detected errors. For software errors, these are the names of resources that have detected errors. For hardware errors, these are names of devices or system components. It does not indicate that the component is faulty or needs replacement. Instead, it is used to determine the appropriate diagnostic modules to be used to analyze the error. The *List* variable values can be separated by , (commas), or enclosed in ″ ″ (double quotation marks) and separated by , (commas) or space characters.
-R *List*	Deletes error-log entries for the resource types specified by the *List* variable. For hardware errors, the *List* variable is a device type. For software errors, the value of the *List* variable is **LPP**. The *List* variable values can be separated by , (commas), or enclosed in ″ ″ (double quotation marks) and separated by , (commas) or space characters.
-S *List*	Deletes error-log entries for the resource classes specified by the *List* variable. For hardware errors, the *List* variable is a device class. The *List* variable values can be separated by , (commas), or enclosed in ″ ″ (double quotation marks) and separated by , (commas) or space characters.
-T *List*	Deletes error-log entries for error types specified by the *List* variable. Valid *List* variable values are: **PERM**, **TEMP**, **PERF**, **PEND**, **INFO**, and **UNKN**. The *List* variable values can be separated by , (commas), or enclosed in ″ ″ (double quotation marks) and separated by, (commas) or space characters.
-y *FileName*	Uses the error-record template file specified by the *FileName* variable.

Security

Access Control: Only the root user can run this command.

Attention RBAC users and Trusted AIX users: This command can perform privileged operations. Only privileged users can run privileged operations. For more information about authorizations and privileges, see Privileged Command Database in *Security*. For a list of privileges and the authorizations associated with this command, see the **/etc/security/privcmds** database file.

Examples

1. To delete all entries from the error log, enter:
   ```
   errclear 0
   ```
2. To delete all entries in the error log classified as software errors, enter:
   ```
   errclear -d S 0
   ```
3. To clear all entries from the alternate error-log file `/var/adm/ras/errlog.alternate`, enter:
   ```
   errclear -i /var/adm/ras/errlog.alternate 0
   ```
4. To clear all hardware entries from the alternate error-log file `/var/adm/ras/errlog.alternate`, enter:
   ```
   errclear -i /var/adm/ras/errlog.alternate -d H 0
   ```

Files

/etc/objrepos/SWservAt	Contains the Software Service Aids Attributes object class, which is the error-log configuration database.

Related Information

The **errdead** command, **errinstall** command, **errlogger** command, **errmsg** command, **errpt** command, **errstop** command, **errupdate** command, **uname** command.

The **errdemon** daemon.

The **errsave** kernel service.

The **errlog** subroutine.

Error Logging Overview in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

For information on installing the Web-based System Manager, see Chapter 2: Installation and System Requirements in *AIX Version 6.1 Web-based System Manager Administration Guide*.

---

errctrl Command

Purpose

Modifies or displays the error-checking attributes of system components. Persistent attribute values can also be specified for components that have not yet been created.

Syntax

errctrl [**-nru**] *ComponentSelector ... subcommand ...*

errctrl -p [**-ru**] *ComponentSelector ... subcommand ...*

errctrl -P [**-ru**] *ComponentSelector ... subcommand ...*

errctrl -x { **-P** | **-p** } [**-ru**] *ComponentSelector ...*

errctrl -q [**-rupP**] {*ComponentSelector ...*}

errctrl {**-h** | **-?**}

errctrl -P {**errcheckon** | **errcheckoff**}

Description

The **errctrl** command modifies or displays the error-checking attribute values of some or all components. Components are selected by name, by alias, or by type or subtype.

The supported value of a *ComponentSelector* is as follows:

-c componentPatternList

-l aliasPatternList

-t typePatternList

Each list consists of one or more patterns separated by quoted spaces or commas. Patterns can contain special characters as described by the **fnmatch** subroutine. The pattern characters question mark (?), asterisk (*), and brackets ([ ]) are supported, but character classes and collation sequences are not allowed inside brackets ([ ]). Specifying **-c all** selects all components, if no other *ComponentSelector* is used.

The **errctrl** command can also be used with the **-p** or **-P** flag to specify persistent attribute customizations. For more information about persistent attribute customizations, see “Persistent Customizations” on page 378.

To enable or disable error checking for all components immediately and persistently, specify the **errcheckon** or **errcheckoff** subcommand with the **-P** flag. No other flags or subcommands are allowed with this form of the command. A **bosboot** command is required to make settings persistent across reboots.

The modified attributes depend on the subcommand specified on the command line. Multiple subcommands can be specified in a single invocation. The following subcommands are available:

errcheckon	Turns on error checking.
errcheckoff	Turns off error checking.
errcheckminimal	Sets the error checking level to 1.
errchecknormal	Sets the error checking level to 3.
errcheckdetail	Sets the error checking level to 7.
errchecklevel={0-9}	Sets the error checking level to the specified value.
lowsevdisposition={disp}	Sets the disposition for low-severity errors to the specified value.
medsevdisposition={disp}	Sets the disposition for medium-severity errors to the specified value.

The **disp** error disposition is one of the following values:

- ignore (or 48)
- log (or 64)
- livedump (or 80)
- isolate (or 96)
- sysdump (or 112)

Other subcommands can be recognized by individual components. A subcommand that is not recognized by a component is ignored.

Current attribute values can be displayed with the **-q** flag. If no *ComponentSelector* is used, attribute values are displayed for all components for which error-checking is supported.

Memory overlay detection system for network memory can be enabled by setting the detailed error level for the netmalloc component. Raise the errlevel for the netmalloc component to five or more (default: 3(normal)) to collect complete network memory police buffer information for all network memory allocation

and free events. Note that raising the error level to seven (detail) or more also enables network memory overlay detection system. To enable only the **net_malloc_police** option and outstanding memory allocation (OSTD) logging for all network memory allocations and free events, raise the error level to five.

For more information about modifying the errlevel, see the “Examples” on page 380. For more information about raising the trace level to collect trace data in the netmalloc component, see the **ctctrl** command.

This command can be used to set the probability (frequency) and values to the following **netmalloc** functions.

- police_frequency
- frag_mask

Probability is the numerator out of 1024 (for example, 10%: 102, 5%: 51, 1%: 10, 0.1%: 1)

Persistent Customizations

The **-p** and **-P** flags allow attribute values to be specified for system components that have not been created yet. Thus, attributes for newly created components can be customized before the components become active. The **-p** flag is used to specify customizations for components that will be created in the future, but before the system is rebooted. The **-P** flag is used to specify customizations that will take effect after the next reboot. These customizations are added to the **/var/adm/ras/raspertune** file. The **bosboot** command must be run to save these customizations in the boot image and the system must be rebooted for the customizations to take effect.

The *ComponentSelectors* can contain pattern-matching characters. Thus, a persistent customization can apply to more than one component. In addition, multiple customizations can apply to the same component, if different *ComponentSelectors* are used. If conflicting attribute values are specified in multiple customizations, the last customization takes precedence. If a customization already exists for a specified *ComponentSelector*, the new customization replaces the old one.

Multiple *ComponentSelectors* are allowed when persistent customizations are specified, but in all cases, using multiple selectors is equivalent to specifying multiple commands, each with a single component selector. For example, the customization `"errctrl -p -l hdisk0 -l hdisk1 errchecknormail"` is equivalent to the following two customizations:

```
errctrl -p -l hdisk0 errchecknormal
errctrl -p -l hdisk1 errchecknormal
```

Customizations specified with the **-p** or **-P** flag are not deleted after they are used. Therefore, a single customization can affect multiple new components. Persistent customization can be deleted with the **-x** flag. The *ComponentSelector* must be specified identically to the way it was specified when the customization was created. For example, if a customization is created with the *ComponentSelector* `-l hdisk0`, the customization cannot be deleted with the *ComponentSelector* `-l hdisk[0]`, even though both *ComponentSelectors* match the same component alias. When a persistent customization is deleted, no change is made to the attributes of components that were created when the customization was active.

Persistent customizations deleted with the **-x** and **-P** flags will remain in effect unless the **bosboot** command is run and the system is rebooted. A persistent customization that was created with the **-P** flag can be deleted after rebooting by using the **-x** and **-p** flags. In this case, the customization will be active again if the system is rebooted.

The **-r** and **-u** flags can be used when specifying persistent customizations. Using one flag specifies a different name space for the specified component selectors. Using both flags at the same time is equivalent to two separate command invocations, each with one of the flags. For example, the persistent customization `"errctrl -p -l hdisk0 -u -r errcheckdetail"` is equivalent to the following two separate customizations:

```
errctrl -p -l hdisk0 -u errcheckdetail
errctrl -p -l hdisk0 -r errcheckdetail
```

The following persistent customizations are all distinct, and can be modified or deleted independently.

```
errctrl -p -l hdisk0 errcheckdetail
errctrl -p -l hdisk0 -r errcheckdetail
errctrl -p -l hdisk0 -u errcheckdetail
```

Persistent customizations can be queried by using the **-q** flag with either the **-P** or **-p** flag. Specifying the **-q** flag with the **-P** flag displays lines from the **/var/adm/ras/raspertune** file. Specifying the **-q** flag with the **-p** and **-r** flags displays the persistent customizations originally specified with the **-r** flag. Without the **-r** flag, the **-q** and **-p** flags display the persistent customizations specified with or without the **-u** flag.

A persistent customization allows multiple subcommands to be specified. If conflicting subcommands are used, the last subcommand is used. For example, the **errchecknormal** and **errcheckdetail** subcommands specify different values for the same error-checking attribute, so the last specified subcommand will be used.

Flags

-c *ComponentList*	Specifies a comma-separated or space-separated list of component names. The **-c all** flag selects all components if it is the only *ComponentSelector*.
-h or **-?**	Displays a usage message.
-l *aliasList*	Specifies a comma-separated or space-separated list of component aliases.
-n	Applies subcommands immediately. This flag is the default if neither the **-p** nor the **-P** flag is used.
-P	Specifies the subcommands that will persist across reboots. You must run the **bosboot** command and reboot for these subcommands to be active.
-x	Deletes the persistent customization for the specified components. The *ComponentSelector* must be entered exactly as they were entered when the customization was originally specified.
-p	Specifies persistent subcommands. The specified subcommands will be applied to newly-created components.
-q	Queries the attribute settings of selected components. This flag can also be used with the **-p** or the **-P** flag to display persistent customizations.
-r	Applies the subcommands recursively to all subcomponents of the selected components.
-t *type_subtypeList*	Specifies a space-separated or a comma-separated list of *type* or *type_subtype* names. Valid *type* names include `device`, `filesystem`, `network`, `services`, `storage`, and `ui`. A complete list of *type* and *type_subtype* names is located in the **/usr/include/sys/ras_base.h** header file.
-u	Applies the subcommands recursively to the ancestors of the specified components.

Note: The **-u** and **-r** flags can be used together. Multiple **-c**, **-l** and **-t** flags can be used on the command line.

Exit Status

0	Successful completion.
>0	An error occurred.

Security

Attention RBAC users and Trusted AIX users: This command can perform privileged operations. Only privileged users can run privileged operations. For more information about authorizations and privileges, see Privileged Command Database in *Security*. For a list of privileges and the authorizations associated with this command, see the **/etc/security/privcmds** database file.

Examples

1. To turn on detailed error checking for all JFS2 userdata components, enter:
   ```
   errctrl -c 'jfs2.filesystem.*.userdata' errcheckdetail
   ```
2. To specify a persistent customization for the userdata component of new JFS2 file systems, enter:
   ```
   errctrl -p -c 'jfs2.filesystem.*.userdata' errcheckminimal
   ```

   The existing userdata components are not affected.
3. To specify a customization that will persist across reboots, enter:
   ```
   errctrl -P -c 'jfs2.filesystem.*.userdata' errcheckminimal
   ```

   If you run the **bosboot** command and the system is rebooted, minimal error checking will be in effect for all JFS2 userdata components.
4. To set minimal error checking for all current and future JFS2 userdata components., enter:
   ```
   errctrl -npP -c 'jfs2.filesystem.*.userdata' errcheckminimal
   ```
5. To specify multiple persistent attribute values for the ethernet component, enter:
   ```
   errctrl -P -c ethernet errcheckminimal medsevdisposition=50
   ```
6. To delete the customization specified in Example 2, enter:
   ```
   errctrl -p -x -c 'jfs2.filesystem.*.userdata'
   ```
7. To list all persistent, recursive-down attribute customizations, enter:
   ```
   errctrl -q -p -r
   ```
8. To list the current error-checking attribute values for the JFS2 components and its descendants, enter:
   ```
   errctrl -q -c jfs2 -r
   ```
9. To enable memory overlay detection system (MODS) for network memory (raise the error level to detailed level for the netmalloc component), enter:
   ```
   errctrl errcheckdetail -c netmalloc
   ```

   or
   ```
   errctrl errchecklevel=7 -c netmalloc
   ```

   Note: This also enables the **net_malloc_police** option for all network memory allocations and free events.
10. To enable the **net_malloc_police** option for all network memory allocations and free events, raise the netmalloc component error level to five or greater, enter:
    ```
    errctrl errchecklevel=5 -c netmalloc
    ```

    This command also enables outstanding memory allocation (OSTD) logging for network memory.
11. To change the frequency of netmalloc police allocation and free events to 25%, change the probability of police_frequency function to 256, enter:
    ```
    errctrl police_frequency=256 –c netmalloc.police
    ```

Location

/usr/sbin/errctrl

Files

/var/adm/ras/raspertune	File containing persistent attribute customizations that will apply after rebooting, if the **bosboot** command is run first.

Related Information

The **ctctrl** and **dumpctrl** commands.

The ras_register and ras_unregister kernel services, ras_customize kernel service, ras_control kernel service.

The /var/adm/ras/raspertune file.

---

errdead Command

Purpose

Extracts error records from a system dump or live dump.

Syntax

/usr/lib/errdead [**-i** *FileName*] *DumpFile* [*UnixFile*]

Description

The **errdead** command extracts error records from a system dump or live dump containing the internal buffer maintained by the **/dev/error** file. The **errdead** command extracts the error records from the dump file and adds those error records directly to the error log.

The error log daemon need not be running when the **errdead** command is run.

Flag

-i *FileName*	Adds the extracted error records to the error log file specified by the *FileName* variable. If the file does not exist, the **errdead** command creates it. If this flag is not specified, the value from the error log configuration database is used.

Parameters

DumpFile	Specifies the dump image to operate on.
UnixFile	Specifies the UNIX file that is in use when the system dump or live dump is taken. This is not necessary if using **errdead** command on the same system that the dump originated from.

Security

Access Control: Only the root user can run this command.

Example

To capture error log information from a dump image that resides in the `/dev/hd7` file, enter:

```
/usr/lib/errdead /dev/hd7
```

Error logging information is in the dump image if the **errdemon** daemon was not running when the dump occurred.

File

/etc/objrepos/SWservAt	Contains the software service aids attributes object class; that is, the error log configuration database.

Related Information

The **errclear** command, **errinstall** command, **errlogger** command, **errmsg** command, **errpt** command, **errstop** command, **errupdate** command.

The **errdemon** daemon.

The **errsave** kernel service.

The **errlog** subroutine.

Error Logging Overview in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*

---

errdemon Daemon

Purpose

Starts error logging daemon (**errdemon**) and writes entries to the error log.

Syntax

errdemon [[**-B** *BufferSize*] [**-d** | **-D**] [**-i** *File*] [**-s** *LogSize*] [**-t** *Time*] [**-m** *MaxDups*] | **-l**]

Description

The error logging daemon reads error records from the **/dev/error** file and creates error log entries in the system error log. Besides writing an entry to the system error log each time an error is logged, the error logging daemon performs error notification as specified in the error notification database. The **/etc/objrepos/errnotify** file is the error notification database. The default system error log is maintained in the **/var/adm/ras/errlog** file. The last error entry is placed in nonvolatile random access memory (NVRAM). During system startup, this last error entry is read from NVRAM and added to the error log when the error logging daemon is started.

The error logging daemon does not create an error log entry for the logged error if the error record template specifies `Log=FALSE`.

If you use the error logging daemon without flags, the system restarts the error logging daemon using the configuration values stored in the error log configuration database. By default, the **errdemon** daemon removes duplicate error log entries when they are logged very rapidly. This is to prevent run away error logging from adversely effecting system performance. The number of duplicate entries can be seen with a detailed error report.

Use the **errclear** command to remove entries from the system error log.

> **Attention:** The error logging daemon is normally started during system initialization. Stopping the error logging daemon can cause error data temporarily stored in internal buffers to be overwritten before it can be recorded in the error log file.

Flags

-B *BufferSize*
Uses the number of bytes specified by the *BufferSize* parameter for the error log device driver's in-memory buffer. The specified buffer size is saved in the error log configuration database. If the *BufferSize* parameter is larger than the buffer size currently in use, the in-memory buffer is immediately increased. If the *BufferSize* parameter is smaller than the buffer size currently in use, the new size is put into effect the next time the error logging daemon is started after the system is rebooted. The buffer cannot be made smaller than the hard-coded default of 8KB.

If this parameter is not specified, the error logging daemon uses the buffer size from the error log configuration database.

The size you specify is rounded up to the next integral multiple of the memory page size (4KB). The memory used for the error log device driver's in-memory buffer is not available for use by other processes. (The buffer is pinned). Be careful not to impact your system's performance by making the buffer excessively large. On the other hand, if you make the buffer too small, the buffer can become full if error entries arrive faster than they can be read from the buffer and put into the log file. When the buffer is full, new entries are discarded until space becomes available in the buffer. When this situation occurs, the error logging daemon creates an error log entry to inform you of the problem. You can correct the problem by enlarging the buffer.

-d
Specifies that duplicate error log entries cannot be removed. The default behavior is to remove duplicates, which is indicated with the **-D** flag.

-D
Specifies that duplicate entries are to be removed. This is the default.

-i *File*
Uses the error log file specified by the *File* variable. The specified file name is saved in the error log configuration database and is immediately put into use.

-l
Displays the values for the error log file name, file size, buffer size, and duplicate handling values from the error log configuration database.

-m *MaxDups*
Specifies the maximum number of duplicate entries allowed before a duplicate error is forced out. The default is 1000. When an error has been duplicated the number of times that is specified in *MaxDups*, a duplicate error is written just as it would be if a unique error was logged. The values allowed for *MaxDups* are 1 to 2147483647.

-s *LogSize*
Uses the size specified by the *LogSize* variable for the maximum size of the error log file. The specified log file size limit is saved in the error log configuration database, and it is immediately put into use. If the log file size limit is smaller than the size of the log file currently in use, the error logging daemon renames the current log file by appending **.old** to the file name. The error logging daemon creates a new log file with the specified size limit. Generate a report from the old log file using the **-i** flag of the **errpt** command.

If this parameter is not specified, the error logging daemon uses the log file size from the error log configuration database.

-t *Time*
Specifies the approximate time interval (in milliseconds) within which an error is considered a duplicate if it is identical to the previous error. Errors occurring after this time interval are not considered duplicates even if they are identical to the previous error. The default interval is 10000, or 10 seconds. The values allowed for Time are 1 to 2147483647.

> **Note:** This flag eliminates duplicate entries in the case of an error logger rapidly logging the same error, this usually indicates a loop condition. It is not intended to catch all duplicate errors for which there may be error notification objects. Making this value sufficiently large may compromise error notification by eliminating too many errors. See the **errpt** command for a description of eliminating duplicate errors in an error report.

Security

Access Control: Only the root user can run this daemon.

Examples

1. To start the error-logging daemon, enter:
   ```
   /usr/lib/errdemon
   ```
2. To view the current maximum error-log size, enter:
   ```
   /usr/lib/errdemon -l
   ```
3. To change the current maximum error-log size from 1MB to 64KB, enter:
   ```
   /usr/lib/errdemon -s 65536
   ```
4. To only consider errors that are logged within the last 10 milliseconds to be duplicates, enter
   ```
   /usr/lib/errdemon -t 10
   ```

Files

/dev/error	Source of error records.
/var/adm/ras/errtmplt	Contains the error template repository.
/usr/lib/errdemon	Contains the **errdemon** daemon.
/etc/objrepos/SWservAt	Contains the software service aids attributes object class; that is, the error log configuration database.

Related Information

The **errclear** command, **errdead** command, **errinstall** command, **errlogger** command, **errmsg** command, **errpt** command, **errstop** command, **errupdate** command.

The **errsave** kernel service.

The **error logging** special files.

The **errlog** subroutine.

Error Logging Overview in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

---

errinstall Command

Purpose

Installs messages in the error logging message sets.

Syntax

errinstall [**-c**] [**-f**] [**-q**] [**-z** *FileName*] *File*

Description

The **errinstall** command is an installation aid that adds or replaces messages in the Error Description, Probable Cause, User Cause, Install Cause, Failure Cause, Recommended Action, and Detailed Data data id message sets of the error log message catalog.

The *File* parameter specifies an input file containing messages to be added or replaced. If you do not specify the *File* parameter or if you specify it as the **-** (minus sign), the **errinstall** command reads from standard input.

> **Note:** Program products and in-house applications should use predefined messages from the error logging message sets. List the predefined messages using the **errmsg -w** command. To add new

messages, third-party software vendors should contact IBM Developer Solutions to register new messages. During the development of in-house applications, the **errmsg** command can be used to add new messages, but the new messages must not conflict with the messages added for other in-house applications.

Undo Feature

The **errinstall** command creates an undo file in the current directory named the *File*.**undo** file. (If the **errinstall** command is reading from standard input, the undo file information is written to standard output.) The *File*.**undo** file can be used as input to the **errinstall** command to undo the changes the **errinstall** command has just made. To undo changes, run the **errinstall** command with the **-f** flag and specify the *File*.**undo** file for the *File* parameter.

Input File (or Standard In) File Format

Two separate lines of information are required to add or replace a single message in the error log message catalog. You can include multiple additions or replacements in a single file. The first line is required to identify the message set to which the message is to be added or replaced. Use the following format:

```
SET MessageSetID
```

where the *MessageSetID* parameter is one of the following single characters:

E Identifies Error Description
P Identifies Probable Cause
U Identifies User Cause
I Identifies Install Cause
F Identifies Failure Cause
R Identifies Recommended Action
D Identifies Detailed Data

The second line lists the message ID with the message to be added or replaced. At least one line is required, and multiple lines can be included, following a single line that identifies a message set. As described earlier, users should contact their service representative to obtain the message ID, unless it is required for an in-house application only (in which case, use the **errmsg** command to install the error message without a predetermined error message ID).

You must put a space between the message ID and the message text, and enclose the text of the message in double quotes as follows:

```
message ID "message text"
```

In addition to the two required lines of information, you can also include lines of comments. A comment line must have a $ (dollar sign) or an * (asterisk) operator in the first column. The asterisk is the preferred choice.

Note: Messages added to the Error Description, Probable Cause, and Detailed Data ID message sets must not exceed 40 characters in length. Messages added to the User Cause, Install Cause, Failure Cause, and Recommended Action message sets must not exceed 128 characters in length. If messages exceed these lengths, the **errinstall** command displays a warning message, but adds the messages to the codepoint catalogue. These messages will be truncated when displayed by the summary errpt command.

Flags

-c Checks the input *File* parameter for syntax errors.

-f Replaces messages having duplicate IDs. When an attempt is made to add a message using a message ID that is already in use, the **-f** flag forces the **errinstall** command to replace the old message text with the new message text. If the **-f** flag is not specified, the old message text is not replaced and a warning message is written to standard error. The **-f** flag is also required to undo a message installation.

-q Suppresses the creation of an **undo** file.

-z *FileName* Uses the error logging message catalog specified by the *FileName* parameter.

Security

Access Control: Only the root user can run this command.

Attention RBAC users and Trusted AIX users: This command can perform privileged operations. Only privileged users can run privileged operations. For more information about authorizations and privileges, see Privileged Command Database in *Security*. For a list of privileges and the authorizations associated with this command, see the **/etc/security/privcmds** database file.

Examples

1. To install the error log messages for the licensed product `lpp`, enter:

   ```
   errinstall -f /tmp/lpp.desc
   ```

2. To undo the changes made to the error log message catalog by the above example of the **errinstall** command, enter:

   ```
   errinstall -f /tmp/lpp.desc.undo
   ```

3. To install an error message in the Probable Cause message set, enter:

   ```
   errinstall

   * Add a probable cause for widget failure:
   SET P
   E100 "widget adapter"
   ```

4. To replace a message with a duplicate ID in the Probable Cause message set, enter:

   ```
   errinstall -f

   * Replace the message associated with ID E100 in the
   * Recommended Action message set
   SET R
   E100 "Replace disk drive"
   ```

5. If you name your input file **in_file** and then want to use it to install new error messages, enter:

   ```
   errinstall in_file
   ```

6. To overwrite existing error messages in message sets, use the previously defined ID numbers in your **in_file**, and specify the **-f** flag with the **errinstall** command as follows:

   ```
   errinstall -f in_file
   ```

7. The following example illustrates sample contents of an input file to be installed.

   ```
   *
   * Add these error messages to the Detailed Data message set:
   *
   SET D
   8105 "Logical channel number"
   8106 "Timer reference stamp"
   *
   * Add these error messages to the Probable Cause message set:
   *
   SET P
   E861 "Bad memory card"
   E865 "Unexpected System Halt"
   E876 "Fiber Optic Cable"
   *
   ```

```
* Add this message to the Recommended Action message set:
*
SET R
E850 "Install updated driver code"
```

Files

/usr/lib/nls/msg/$LANG/codepoint.cat	Contains the error log message catalog. In the United States, the value of the **$LANG** environment variable is **En_US**.

Related Information

The **errclear** command, **errdead** command, **errlogger** command, **errmsg** command, **errpt** command, **errstop** command, **errupdate** command.

The **errdemon** daemon.

The **errsave** kernel service.

The **errlog** subroutine.

The **error logging** special files.

TheError Logging Overview in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*

---

errlogger Command

Purpose

Logs an operator message.

Syntax

errlogger *Message*

Description

The **errlogger** command creates an operator error log entry that contains an operator message up to 1024 bytes in length.

Security

Access Control: Only the root user can run this command.

Attention RBAC users and Trusted AIX users: This command can perform privileged operations. Only privileged users can run privileged operations. For more information about authorizations and privileges, see Privileged Command Database in *Security*. For a list of privileges and the authorizations associated with this command, see the **/etc/security/privcmds** database file.

Examples

To create an operator message for system drive reconfiguration, enter:

```
errlogger system drive reconfigured
```

Related Information

The **errpt** command.

The **errsave** kernel service.

The **errlog** subroutine.

The Error Logging Overview in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*

---

errmsg Command

Purpose

Adds a message to the error log message catalog.

Syntax

errmsg [**-c**] [**-z** *FileName*] [**-w** *Set_List* | *File*]

Description

The **errmsg** command updates and displays the error-log message catalog containing the Error Description, Probable Cause, User Cause, Install Cause, Failure Cause, Recommended Action, and Detailed Data ID message sets.

The message sets to which messages are to be added or deleted are listed in the input *File* parameter as follows:

***** or **$**	Comment lines must have an * (asterisk) or $ (dollar sign) comment operator in the first column. The * is the preferred choice.
+	Messages to be added must be preceded by a + (plus sign).
-	Messages to be deleted must be preceded by a - (minus sign).
SET	Message set ID.
″**Message Text**″	Message text must be enclosed in double quotation marks.
Message ID	Message ID of the message to be deleted.

Messages added to the Error Description, Probable Cause, and Detailed Data ID message sets must not exceed 40 characters in length. Messages added to the User Cause, Install Cause, Failure Cause, and Recommended Action message sets must not exceed 128 characters in length. A maximum of 2047 user-defined messages can be added to each message set.

The **errmsg** command is used by application developers to create new messages used in the Error Record Templates Repository. An existing message should always be used, if possible.

If no flags are specified on the command line, the default operation is an update. Updates are specified in the input *File* parameter. If the input *File* parameter is not specified or if a **-** (minus sign) is specified instead of the *File* parameter, the **errmsg** command reads from standard input. For each message that is added, the **errmsg** command assigns an identifier. In addition to adding the message to the message catalog, the **errmsg** command writes the identifier and message text to the *File***.out** file. The *File***.out** file is also created when deletions are made from the message catalog. If the **errmsg** command is reading from standard input, the identifier and message text are written to standard output.

Flags

-c Checks the input file for syntax errors.

-w *Set_List* Displays the error log message sets specified by the *Set_List* variables. This option displays the messages contained in the Error Log message sets and their identifiers. Output is written to standard output. The *Set_List* variables can be separated by commas or enclosed in double-quotation marks and separated by commas or blanks. The *Set_List* variables are the message set IDs or, if the value of the *Set_List* variable **all** is specified, the contents of all of the Error Log message sets are displayed. The valid values of the *Set_List* variables are:

- **all** Displays all message sets
- **D** Displays Detailed Data ID message set
- **E** Displays Error Description message set
- **F** Displays Failure Cause message set
- **I** Displays Install Cause message set
- **P** Displays Probable Cause message set
- **R** Displays Recommended Action message set
- **U** Displays User Cause message set

-z *Filename* Uses the error-logging message catalog specified by the *Filename* variable.

Security

Access Control: Only the root user can run this command.

Attention RBAC users and Trusted AIX users: This command can perform privileged operations. Only privileged users can run privileged operations. For more information about authorizations and privileges, see Privileged Command Database in *Security*. For a list of privileges and the authorizations associated with this command, see the **/etc/security/privcmds** database file.

Examples

1. To delete messages from the Probable Cause message set, enter:

```
errmsg
* Delete messages FF1A, FF1B, and FF1C from the Probable Cause
* message set
SET P
- FF1A
- FF1B
- FF1C
```

2. To add a message to the Probable Cause message set for the Widget Failure error, enter:

```
errmsg
* Add a Probable Cause for Widget Failure
SET P
+ "WIDGET ADAPTER"
```

File

/usr/lib/nls/msg/$LANG/codepoint.cat Contains the error log message catalog. In the United States, the value of **$LANG** is **En_US**.

Related Information

The **errclear** command, **errdead** command, **errinstall** command, **errlogger** command, **errpt** command, **errstop** command, **errupdate** command.

The **errdemon** daemon.

The **errsave** kernel service.

The **errlog** subroutine.

The **error logging** special files.

Error Logging Overview in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*

---

errpt Command

Purpose

Generates a report of logged errors.

Syntax

To Process a Report from the Error Log

errpt [**-a**] [**-A**] [**-c**] [**-d** *ErrorClassList*] [**-D**] [**-e** *EndDate*] [**-g**] [**-i** *File*] [**-l** *File*] [**-j** *ErrorID* [**,***ErrorID*]] | [**-k** *ErrorID* [**,***ErrorID*]] [**-J** *ErrorLabel* [**,***ErrorLabel*]] | [**-K** *ErrorLabel* [**,***ErrorLabel*]] [**-l** *SequenceNumber*] [**-m** *Machine*] [**-n** *Node*] [**-s** *StartDate*] [**-F** *FlagList*] [**-N** *ResourceNameList*] [**-P**] [**-R** *ResourceTypeList*] [**-S** *ResourceClassList*] [**-T** *ErrorTypeList*] [**-y** *File*] [**-z** *File*]

To Process a Report from the Error Record Template Repository

errpt [**-a**] [**-A**] [**-l** *File*] [**-t**] [**-d** *ErrorClassList*] [**-j** *ErrorID* [**,***ErrorID*]] | [**-k** *ErrorID* [**,***ErrorID*]] [**-J** *ErrorLabel* [**,***ErrorLabel*]] |
[**-K** *ErrorLabel* [**,***ErrorLabel*]] [**-F** *FlagList*] [**-P**] [**-T** *ErrorTypeList*] [**-y** *File*] [**-z** *File*]

Description

The **errpt** command generates an error report from entries in an error log. It includes flags for selecting errors that match specific criteria. By using the default condition, you can display error log entries in the reverse order they occurred and were recorded. By using the **- c** (concurrent) flag, you can display errors as they occur. If the **-i** flag is not used with the **errpt** command, the error log file processed by **errpt** is the one specified in the error log configuration database. (To view the information in the error log configuration database, use the **errdemon** command.)

The default summary report contains one line of data for each error. You can use flags to generate reports with different formats.

> **Note:** The **errpt** command does not perform error log analysis; for analysis, use the **diag** command. When error log analysis is performed, however, diagnostics may add diagnostic information back into the error log. Such information is shown following the detailed data of the corresponding error log entry.

You can use the Devices application in Web-based System Manager (wsm) to change device characteristics. You could also use the System Management Interface Tool (SMIT) **smit errpt** fast path to run this command.

Flags

-a Displays information about errors in the error log file in detailed format. If used in conjunction with the **-t** flag, all the information from the template file is displayed.

-A Displays a shortened version of the detailed report produced by the **-a** flag. The **-A** flag is not valid with the **-a**, **-g**, or **-t** flags. The items reported are the label, date and time, type, resource name, description, and detail data. The example output of this flag is in the following format:

```
LABEL:           STOK_RCVRY_EXIT
Date/Time:       Tue Dec 14 15:25:33
Type:            TEMP
Resource Name:   tok0
Description
PROBLEM RESOLVED
Detail Data
FILE NAME
line: 273 file: stok_wdt.c
SENSE DATA
0000 0000 0000 0000 0000 0000
DEVICE ADDRESS
0004 AC62 25F1
```

-c Formats and displays each of the error entries concurrently, that is, at the time they are logged. The existing entries in the log file are displayed in the order in which they were logged.

-d *ErrorClassList* Limits the error report to certain types of error records specified by the valid *ErrorClassList* variable: **H** (hardware), **S** (software), **0** (**errlogger** command messages), and **U** (undetermined). The error records in the *ErrorClassList* variable can be separated by a **,** (comma), or enclosed in " " (double quotation marks) and separated by a **,** (comma), or a space character.

-D Consolidates duplicate errors. The detailed error report, obtained with the **-a** flag, reports the number, and first and last times of the duplicates. See Error Logging Overview in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

Note: The **-D** flag is not valid with the **-c**, **-g**, **-l**, **-t**, and **-P** flags.

-e *EndDate* Specifies all records posted prior to and including the *EndDate* variable, where the *EndDate* variable has the form *mmddhhmmyy* (month, day, hour, minute, and year).

-g Displays the ASCII representation of unformatted error-log entries. The output of this flag is in the following format:

el_sequence
Error-log stamp number

el_label
Error label

el_timestamp
Error-log entry time stamp

el_crcid
Unique cyclic-redundancy-check (CRC) error identifier

el_machineid
Machine ID variable

el_nodeid
Node ID variable

el_class
Error class

el_type
Error type

el_resource
Resource name

el_rclass
Resource class

el_rtype
Resource type

el_vpd_ibm
IBM vital product data (VPD)

el_vpd_user
User VPD

el_in Location code of a device

el_connwhere
Hardware-connection ID (location on a specific device, such as slot number)

et_label
Error label

et_class
Error class

et_type
Error type

et_desc
Error description

et_probcauses
Probable causes

et_usercauses
User causes

et_useraction
User actions

et_instcauses
Installation causes

et_instaction
Installation actions

et_failcauses
Failure causes

et_failaction
Failure actions

et_detail_length
Detail-data field length

et_detail_descid
Detail-data identifiers

et_detail_encode
Description of detail-data input format

et_logflg
Log flag

et_alertflg
Alertable error flag

et_reportflg
Error report flag

el_detail_length
Detail-data input length

el_detail_data
Detail-data input

-F *FlagList*
Selects error-record templates according to the value of the `Alert`, `Log`, or `Report` field of the template. The *FlagList* variable can be separated by a **,** (comma), or enclosed in " " (double quotation marks) and separated by a **,** (comma), or a space character. The **-F** flag is used with the **-t** flag only.

Valid values of the *FlagList* variable include:

alert=0 Selects error-record templates with the `Alert` field set to False.

alert=1 Selects error-record templates with the `Alert` field set to True.

log=0 Selects error-record templates with the `Log` field set to False.

log=1 Selects error-record templates with the `Log` field set to True.

report=0
Selects error-record templates with the `Report` field set to False.

report=1
Selects error-record templates with the `Report` field set to True.

-i *File*
Uses the error log file specified by the *File* variable. If this flag is not specified, the value from the error log configuration database is used.

-l *File*
Uses the diagnostic log file specified by *File*. If this flag is not specified, the default pathname, **/var/adm/ras/diag_log**, is used

-j *ErrorID*[**,***ErrorID*]
Includes only the error-log entries specified by the *ErrorID* (error identifier) variable. The *ErrorID* variables can be separated by a **,** (comma), or enclosed in " " (double quotation marks) and separated by a **,** (comma), or a space character. When combined with the **-t** flag, entries are processed from the error-template repository. (Otherwise entries are processed from the error-log repository.)

-J *ErrorLabel* Includes the error log entries specified by the *ErrorLabel* variable. The *ErrorLabel* variable values can be separated by commas or enclosed in double-quotation marks and separated by commas or blanks. When combined with the **-t** flag, entries are processed from the error template repository. (Otherwise, entries are processed from the error log repository.)

-k *ErrorID*[**,***ErrorID*] Excludes the error-log entries specified by the *ErrorID* variable. The *ErrorID* variables can be separated by a **,** (comma), or enclosed in ″ ″ (double quotation marks) and separated by a **,** (comma), or a space character. When combined with the **-t** flag, entries are processed from the error-template repository. (Otherwise entries are processed from the error-log repository.)

-K *ErrorLabel* Excludes the error log entries specified by the *ErrorLabel* variable. The *ErrorLabel* variable values can be separated by commas or enclosed in double-quotation marks and separated by commas or blanks. When combined with the **-t** flag, entries are processed from the error template repository. (Otherwise, entries are processed from the error log repository).

-l *SequenceNumber* Selects a unique error-log entry specified by the *SequenceNumber* variable. This flag is used by methods in the error-notification object class. The *SequenceNumber* variable can be separated by a **,** (comma), or enclosed in ″ ″ (double quotation marks) and separated by a **,** (comma), or a space character.

-m *Machine* Includes error-log entries for the specified *Machine* variable. The **uname -m** command returns the *Machine* variable value.

-n *Node* Includes error-log entries for the specified *Node* variable. The **uname -n** command returns the *Node* variable value.

-N *ResourceNameList* Generates a report of resource names specified by the *ResourceNameList* variable. The *ResourceNameList* variable is a list of names of resources that have detected errors. For software errors, the *ResourceNameList* variable lists the names of resources that have detected errors. For hardware errors, it lists names of devices or system components. It does not indicate that the component is faulty or needs replacement. Instead, it is used to determine the appropriate diagnostic modules to be used to analyze the error.

The names of the *ResourceNameList* variable can be separated by a **,** (comma), or enclosed in ″ ″ (double quotation marks) and separated by a **,** (comma), or a space character.

-P Shows only errors which are duplicates of the previous error. The **-P** flag applies only to duplicate errors generated by the error log device driver. These errors are duplicates that occurred within the approximate time interval specified by the **errlg_duptime** error logging attribute controlled by the **errdemon** daemon **-t** flag. The **-P** flag is invalid with the **-D** flag.

-R *ResourceTypeList* Generates a report of resource types specified by the *ResourceTypeList* variable. For hardware errors, the *ResourceTypeList* variable is a device type. For software errors, it is the **LPP** value. The items in the *ResourceTypeList* variable can be each separated by a **,** (comma), or enclosed in ″ ″ (double quotation marks) and separated by a **,** (comma), or a space character.

-s *StartDate* Specifies all records posted on and after the *StartDate* variable, where the *StartDate* variable has the format *mmddhhmmyy* (month, day, hour, minute, and year).

-S *ResourceClassList* Generates a report of resource classes specified by the *ResourceClassList* variable. For hardware errors, the *ResourceClassList* variable is a device class. The resource classes must be each separated by a **,** (comma), or enclosed in ″ ″ (double quotation marks) and separated by a **,** (comma), or a space character.

-t Processes the error-record template repository instead of the error log. The **-t** flag can be used to view error-record templates in report form.

-T *ErrorTypeList* Limits the error report to error types specified by the valid *ErrorTypeList* variables: **INFO**, **PEND**, **PERF**, **PERM**, **TEMP**, and **UNKN**. The error types can be each separated by a **,** (comma), or enclosed in ″ ″ (double quotation marks) and separated by a **,** or a space character.

-y *File* Uses the error record template file specified by the *File* variable. When combined with the **-t** flag, entries are processed from the specified error template repository. (Otherwise, entries are processed from the error log repository, using the specified error template repository.)

-z *File* Uses the error logging message catalog specified by the *File* variable. When combined with the **-t** flag, entries are processed from the error template repository. (Otherwise, entries are processed from the error log repository.)

Examples

1. To display a complete summary report, enter:
   ```
   errpt
   ```
2. To display a complete detailed report, enter:
   ```
   errpt  -a
   ```
3. To display a detailed report of all errors logged for the error identifier `E19E094F`, enter:
   ```
   errpt  -a  -j E19E094F
   ```
4. To display a detailed report of all errors logged in the past 24 hours, enter:
   ```
   errpt  -a  -s mmddhhmmyy
   ```
   where the `mmddhhmmyy` string equals the current month, day, hour, minute, and year, minus 24 hours.
5. To list error-record templates for which logging is turned off for any error-log entries, enter:
   ```
   errpt  -t  -F log=0
   ```
6. To view all entries from the alternate error-log file `/var/adm/ras/errlog.alternate`, enter:
   ```
   errpt  -i /var/adm/ras/errlog.alternate
   ```
7. To view all hardware entries from the alternate error-log file `/var/adm/ras/errlog.alternate`, enter:
   ```
   errpt  -i /var/adm/ras/errlog.alternate -d H
   ```
8. To display a detailed report of all errors logged for the error label `ERRLOG_ON`, enter:
   ```
   errpt  -a  -J ERRLOG_ON
   ```
9. To display a detailed report of all errors and group duplicate errors, enter:
   ```
   errpt -aD
   ```
10. To display a detailed report of all errors logged for the error labels `DISK_ERR1` and `DISK_ERR2` during the month of August, enter:
    ```
    errpt -a -J DISK_ERR1,DISK_ERR2 -s 0801000004 -e 0831235904"
    ```

Files

/etc/objrepos/SWservAt	Contains the software service aids attributes object class; that is, the error log configuration database.

Related Information

The **diag** command, **errclear** command, **errinstall** command, **errupdate** command, **uname** command.

The **errsave** kernel service.

The **errlog** subroutine.

Error Logging Overview in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*

Examples of Detailed Error Reports, Example of a Summary Error Report in *AIX Version 6.1 General Programming Concepts*.

For information on installing the Web-based System Manager, see Chapter 2: Installation and System Requirements in *AIX Version 6.1 Web-based System Manager Administration Guide*.

errstop Command

Purpose

Terminates the error logging daemon.

Syntax

errstop

Description

> **Attention:** Running the **errstop** command disables diagnostic and recovery functions. Normally the **errdemon** command is started automatically during system initialization and stopped during system shutdown. The error log should never be stopped during normal operations. The **errstop** command should only be used during special circumstances when it is absolutely required and the consequences are clearly understood.

The **errstop** command stops the error logging daemon initiated by the **errdemon** command.

Security

Access Control: Only a root user can run this command.

Examples

To terminate the **errdemon** daemon, enter:

```
/usr/lib/errstop
```

Related Information

The **errclear** command, **errdead** command, **errinstall** command, **errlogger** command, **errmsg** command, **errpt** command, **errupdate** command.

The **errdemon** daemon.

The **errsave** kernel service.

The **errlog** subroutine.

The Error Logging Overview in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*

errupdate Command

Purpose

Updates the Error Record Template Repository.

Syntax

errupdate [**-c**] [**-f**] [**-h**] [**-n**] [**-p**] [**-q**] [**-y***FileName*] [*File*]

Description

The **errupdate** command adds or deletes entries in the Error Record Template Repository, or modifies the log, report, or alert characteristics of existing entries. The **errupdate** command reads from the specified *File* parameter. If the *File* parameter is not specified, the **errupdate** command reads from standard input and writes to standard output.

Each entry to be added, deleted, or modified must be preceded by an operator. The valid operators are:

+	Adds an entry (add operator).
-	Deletes an entry (delete operator).
=	Modifies the log, report, or alert characteristics of an entry.

Entries in the input file must be separated by a blank line.

Comments in the input file can be placed between templates and are indicated by an * (asterisk) in the first column.

If X/Open Portability Guide Issue 4 messages are used in error templates, a message catalog must be specified. This can be done with a line of the form:

```
<*!catalog-name>
```

For example

```
*!mycat.cat
```

The catalog specified applies to XPG4 messages found in subsequent templates, until another ″*!″ catalog specifier is encountered. Also, the ″*!″ specifier may be overridden on an individual template basis with the ″catname″ keyword.

Unless a full pathname to the catalog is specified, the normal rules for retrieving a message catalog are followed. For example, in the above example, mycat.cat is assumed to be in **/usr/lib/nls/msg/%L**.

Entries to be added must be defined in a specific format. The general form of the error record template is:

```
Error Record Template

        + LABEL:
                        Comment=
                        Class=
                        Log=
                        Report=
                        Alert=
                        Err_Type=
                        Err_Desc=
                        Prob_Causes=
                        User_Causes=
                        User_Actions=
                        Inst_Causes=
                        Inst_Actions=
                        Fail_Causes=
                        Fail_Actions=
                        Detail_Data= <data_len>, <data_id>,
                        <data_encoding>
```

Additionally, a catalog name for XPG4 messages can be specified with:

```
catname = <catalog>
```

Any template which contains XPG4 messages, the catname keyword, more than eight detail data items will be referred to as an XPG4 template. An XPG4 template is not alertable, and uses a slightly different calculation for the error id.

The error record template fields are described as follows:

`Alert` Indicates that the error log entry can be processed by products that conform to the SNA Generic Alert Architecture. The `Alert` field can be set to True or False. If this field is omitted from the template, its value will default to False. If the `Alert` field is set to True, the **errupdate** command does not add the template unless the contents of the `Err_Desc` , `Inst_Actions` , `Fail_Cause`, `Fail_Actions` , and `Detail_Data data_id` fields are values recognized by the SNA Generic Alert Architecture (in publication GA27-3136). If any of the values used are not recognized by the SNA Generic Alert Architecture or the template is an XPG4 template, and the `Alert` field is set to True, the **-p** flag must be specified to add or update the template.

`Class` Describes whether the error occurred in hardware or software, is an operator message, or is undetermined. One of the following class descriptors must be specified:

H Indicates the error is a hardware failure.

O Indicates the error is an operator message.

S Indicates the error is a software failure.

U Indicates the error is undetermined.

`Comment` Specifies a comment to be included with the **#define** statement that was created for the Error ID message set. The comment must not exceed 40 characters and must be enclosed in double quotation marks. Comments longer than 40 characters are automatically truncated. The **errupdate** command encloses the comment in the C language comment delimiters, /* (slash, asterisk) and */ (asterisk, slash).

Detail_Data

Describes detailed data, such as detecting module names, sense data, or return codes, that are logged with the error when the error occurs. If no detailed data is logged with the error, this field can be left blank or it can display a message from the Detailed Data ID message set by specifying a **data_len** value of zero. The following three values are required for each **Detail_Data** field and must be separated by commas:

data_len
Number of bytes of data to be associated with the **data_id** value. The **data_len** value is interpreted as a decimal value. To specify environment dependent size, use ″W″. ″W″ will be treated as 8 bytes if error is logged from a 64-bit environment, otherwise 4 bytes.

Note: During detail data length calculation, each ″W″ is treated as 8 bytes long, and it is not case sensitive.

data_id
Identifies a text message from the Detailed Data ID message set “D” to be printed in the error report in front of the detailed data. The value is interpreted as an unsigned hexadecimal up to 4 digits in length.

data_encoding
Describes how detailed data is to be printed in an error report. Valid values are:

ALPHA
The detailed data is a printable ASCII character string.

DEC The detailed data is the binary representation of an integer value, and the decimal equivalent is to be printed.

LDEC The detailed data is the binary representation of a 64-bit value, and the decimal equivalent is to be printed.

HEX The detailed data is to be printed in hexadecimal.

Up to 16 Detail_Data entries may be specified per template. The amount of data logged with an error must not exceed ERR_REC_MAX defined in the **/usr/include/sys/err_rec.h** file. Error data that cannot be contained in an error log entry should be saved elsewhere. Detailed data in the error log entry should contain information that can be used to correlate the error data and the error log entry.

Err_Desc

Describes the error that has occurred. An Error Description message identifier must be specified in this field. This value identifies a text message from the Error Description message set “E” to be displayed for an occurrence of the error. The value is interpreted as an unsigned hexadecimal up to 4 digits in length. The field may also specify an XPG4 style message. This is discussed later.

Err_Type

Describes the severity of the error that has occurred. One of the following values must be specified:

PERF Condition where the performance of the device or component has degraded to below an acceptable level (performance).

PERM Condition that cannot be recovered from (permanent).

PEND Condition signifying that the loss of availability of a device or component is imminent (impending).

TEMP Condition that was recovered from after a number of unsuccessful attempts (temporary).

UNKN Condition where it is not possible to determine the severity of the error (unknown).

INFO Condition for informational error log entry.

Fail_Actions
: Describes recommended actions for correcting an error that resulted from a failure cause. A list of up to 4 Recommended Action message identifiers separated by commas can be specified. This value identifies a text message from the Recommended Action message set “R” to be displayed for an occurrence of the error. The value is interpreted as an unsigned hexadecimal up to four digits in length. This field must be blank if the Fail_Causes field is blank.

  The order in which the recommended actions are listed should be determined by the expense of the action and the probability that the action will correct the error. Always list the actions that have little or no cost (or little or no impact) on the system first. List the actions for which the probability of correcting the error is equal or nearly equal next, with the least expensive actions first. List the remaining actions in order of decreasing probability. The field may also specify an XPG4 style message. This is discussed later.

Fail_Causes
: Describes failure causes for the error that has occurred. A failure cause is defined as a condition that resulted from the failure of a resource. This field can list up to four Failure Cause message identifiers separated by commas. This value identifies a text message from the Failure Cause messages set “F” to be displayed for an occurrence of the error. The value is interpreted as an unsigned hexadecimal up to 4 digits in length. List the failure causes in order of decreasing probability. This field can be left blank if it does not apply to the error that has occurred. If this field is blank, either the User_Causes or the Inst_Causes field must not be blank. The field may also specify an XPG4 style message. This is discussed later.

Inst_Actions
: Describes recommended actions for correcting an install caused error. This field can list of up to 4 Recommended Action message identifiers separated by commas. This value identifies a text message from the Recommended Action message set “R” to be displayed for an occurrence of the error. The value is interpreted as an unsigned hexadecimal up to four digits in length. This field must be blank if the Inst_Causes field was left blank. The order in which the recommended actions are listed is determined by the expense of the action and the probability that the action will correct the error. The actions that have little or no cost or little or no impact on the system should always be listed first. Actions for which the probability of correcting the error are equal or nearly equal should be listed next, with the least expensive actions first. The remaining actions should be listed in order of decreasing probability. The field may also specify an XPG4 style message. This is discussed later.

Inst_Causes
: Describes install causes for the error that has occurred. An install cause is defined to be a condition that resulted from the initial installation or setup of a resource. A list of up to 4 Install Cause message identifiers separated by commas can be specified. This value identifies a text message from the Install Cause message set “I” to be displayed for an occurrence of the error. The value is interpreted as an unsigned hexadecimal up to four digits in length. Install causes should be listed in order of decreasing probability. This field can be left blank if it is not applicable to the error that has occurred. If this field is left blank, the User_Causes or the Fail_Causes field must be non-blank. The field may also specify an XPG4 style message. This is discussed later.

LABEL
: Specifies a unique label of up to 19 characters that must be provided for each error logging template. A string containing “ #define #ERRID_label Error_ID ”, where the **Error_ID** value is the unique ID assigned to the Error Record Template is written to standard output if the **-h** flag was specified at the command line.

Log
: Specifies whether an error log entry should be created for this error when it occurs. The log field can be set to True or False. If this field is omitted from the template, its value will default to True. When this field is set to False, the Report and Alert fields are ignored.

Prob_Causes
: Describes 1 or more probable causes for the error that has occurred. A list of up to 4 Probable Cause message identifiers separated by commas can be specified. This value identifies a text message from the Probable Cause message set “P” to be displayed for an occurrence of the error. The value is interpreted as an unsigned hexadecimal up to 4 digits in length. Probable causes should be listed in order of decreasing probability. At least one probable cause is required. The field may also specify an XPG4 style message. This is discussed later.

Report
: Specifies whether logged occurrences of this error should be reported when an error report is printed. The Report field can be set to True or False. If this field is omitted from the template, its value will default to True.

User_Actions	Describes recommended actions for correcting a user-caused error. A list of up to 4 Recommended Action message identifiers separated by commas can be specified. This value identifies a text message from the Recommended Action message set “R” to be displayed for an occurrence of the error. The value is interpreted as an unsigned hexadecimal up to 4 digits in length. This field must be left blank if the User_Causes field was left blank. The order in which the recommended actions are listed is determined by the expense of the error and the probability that the action will correct the error. The actions that have little or no cost, or little or no impact on the system should always be listed first. Actions for which the probability of correcting the error are equal or nearly equal should be listed next, with the least expensive actions first. The remaining actions should be listed in order of decreasing probability. The field may also specify an XPG4 style message. This is discussed later.
User_Causes	Describes user causes for the error that has occurred. A user cause is defined as a condition that can be corrected without contacting a service organization. A list of up to four User Cause message identifiers separated by commas can be specified. This value identifies a text message from the User Cause message set “U” to be displayed for an occurrence of the error. The value is interpreted as an unsigned hexadecimal up to four digits in length. User causes should be listed in order of decreasing probability. This field can be left blank if it is not applicable to the error that has occurred. If this field is left blank, the Inst_Causes or the Fail_Causes field must be non-blank. The field may also specify an XPG4 style message. This is discussed later.

The catname is used to specify a message catalog to be used for retrieving XPG4 messages for the current template. This will override a catalog specified with a previous ″*!″ catalog specifier. Any template containing XPG4 messages must have a catalog specified either with catname or ″*!″. The catalog name must be enclosed in quotes. Unless a full pathname to the catalog is specified, the normal rules for retrieving a message catalog are followed.

For example, if

```
catname = "mycat.cat"
```

is specified, mycat.cat is assumed to be in **/usr/lib/nls/msg/%L**.

The Error Description, Probable Cause, User Cause, Install Cause, Failure Cause, Recommended Actions, and Detailed Data ID messages must be either an error message identifier maintained in the error log message catalog, or an XPG4 message.

An error message identifier consists of up to 4 hexadecimal digits, without any leading ″0x″. For example, 1234 or ABCD. The **errmsg -w** command can be used to print these messages along with their identifiers. The **errmsg** command can be used to add new messages.

An XPG4 message is specified using the form

```
{<set>, <number>, <"default text">}
```

The set, number, and default text are all required. Symbolic message references are not supported. Also, templates which contain XPG4 messages are not alertable.

A message catalog must be specified for XPG4 messages. This is done with either the ″*!″ catalog specifier, or the catname keyword.

Error logging does not support all the features of normal error messaging. Strings used in error log templates must conform to some restrictions.

- Variable substitution is not supported. For example, the strings may not be used as format specifiers to print values. The strings may only contain the formatting characters ″\t″ and ″\n″.
- The default text strings may not be longer than 1 kb, 1024 bytes.

- It must be noted that the error description is printed in a 40 character area on the non-detailed reports. No string formatting is done for these reports, and only the first 40 characters will be printed.
- The strings should not contain a trailing new line. This is supplied by errpt.

For each entry added, the errupdate command assigns a unique Error ID that is written to the header file specified by *File*.h (where the *File* parameter is the name of the errupdate command input file). If the errupdate command is reading from standard input, the `#define` statement is written to standard output. The values supplied for the `Class` , `Err_Desc` , `Err_Type` , `Fail_Actions` , `Fail_Causes` , `Inst_Actions` , `Inst_Causes` , `Prob_Causes` , `User_Actions` , `User_Causes` fields, and the `Detail_Data` . `data_id` value, are used to calculate the unique Error ID for that error. For XPG4 templates, the Label is also included in the calculation.

The contents of the `Log` , `Report` , and `Alert` fields are not included in the calculation of the unique Error ID; therefore, the log, report, and alert characteristics of a particular error can be modified at any time in the error entry definition stored in the Error Record Template Repository using the **errupdate** command. Also note that the data_len and data_encode portions of the detail data field are not used.

The **errupdate** command also creates an undo file in the current directory named *File*.**undo**. If the **errupdate** command is reading from standard input, the **undo** file is written to **errids.undo** file. The **undo** file contains inputs to the **errupdate** command to undo changes the **errupdate** command has made.

The **errpt -t** command can be used to view the contents of the Error Record Template Repository. The templates are processed and printed as they would appear in an actual error report.

> **Attention:** If you change the error templates be aware that these templates may be changed by a subsequent update. You should keep a record of all changes made and re-apply the changes when your system is updated. This is usually only necessary after a major system update such as moving to a new level of the operating system. Also, such a record allows you to change your templates if you re-install. The easiest way to keep such a record is to always make your template modifications from one errupdate source file.

Flags

-c	Checks the input file for syntax errors.
-f	Forces all templates to be updated, including any templates with error ids identical to ones in the input templates
-h	Creates a `#define` statement for each Error ID assigned to an error template. If a file name was supplied on the command line, the header file name will be that supplied file name appended with **.h**. Otherwise, the `#define` statements are written to standard output.
-n	Suppresses the addition of the error record template to the Error Record Template Repository.
-p	Adds or updates a template with the `Alert` field set to True that contains Error Description, Probable Cause, User Cause, User Action, Install Cause, Install Action, Failure Cause, Fail Action, or Detailed Data data id values that are not recognized by the SNA Generic Alert Architecture (in publication GA27-3136). The **errupdate** command will not let you add a template with these characteristics unless you specify this flag.
-q	Suppresses the creation of an undo file.
-y *FileName*	Uses the error record template file specified by the *FileName* parameter.

Security

Access Control: None, but you must have write authority to a template file you're changing, /var/adm/ras/errtmplt by default.

Attention RBAC users and Trusted AIX users: This command can perform privileged operations. Only privileged users can run privileged operations. For more information about authorizations and privileges,

see Privileged Command Database in *Security*. For a list of privileges and the authorizations associated with this command, see the **/etc/security/privcmds** database file.

Examples

1. To add an entry, define the entry in the input file in the following manner:

```
+ CDROM_ERR22:
        Comment=        "Temporary CDROM read error"
        Class=  H
        Log=            True
        Report= True
        Alert=          False
        Err_Type=       TEMP
        Err_Desc=       E801
        Prob_Causes=    5004
        Fail_Causes=    E800, 6312
        Fail_Actions=   1601, 0000
        Detail_Data=    120, 11, HEX
        Detail_Data=    4, 8058, DEC
        Detail_Data=    4, 8059, DEC
```

   To enter the data,

```
errupdate <input file>
```

2. To modify the log, report, and alert characteristics of entry `99999999` , specify the modify operator = (equal sign) followed by the unique Error ID, and the new characteristics for the entry to be modified:

```
errupdate
 =99999999:
 Report = False
 Log = True
```

3. To delete entry `99999999` from the Error Record Template Repository, specify the delete operator - (minus sign) followed by the unique Error ID of the entry to be deleted:

```
errupdate
 -99999999:
```

4. To override the XPG4 message catalog specified for this input stream with ″*!″, use the ″catname″ keyword.

```
*!mycat.cat
```

   * `mycat.cat` is used for all XPG4 messages from now on.

   * except for this one:

```
+ CDROM_ERR23:
         Comment=        "Temporary CDROM read error"
         catname= "othercat.cat"
         Class=  H
         Log=            True
         Report= True
         Alert=          False
         Err_Type=       TEMP
         Err_Desc=       {1, 1, "CD ROM is broken"}
         Prob_Causes=    {2, 1, "cause 1"},\
                        {2, 2, "Cause 2"}
         Fail_Causes=    E800, 6312
         Fail_Actions=   1601, 0000
         Detail_Data=    120, 11, HEX
         Detail_Data=    4, 8058, DEC
         Detail_Data=    4, 8059, DEC
```

The catalog othercat.cat will be used for the CDROM_ERR23 template only.

Note: A template may contain both XPG4 messages and the traditional error ids or codepoints.

Files

/usr/include/sys/errids.h	Contains the header file that contains Error IDs.
/usr/include/sys/err_rec.h	Contains the header file that contains structures for logging errors.

Related Information

The **errclear** command, **errdead** command, **errinstall** command, **errlogger** command, **errmsg** command, **errpt** command, **errstop** command.

The **errdemon** daemon.

The **errsave** kernel service.

The **errlog** subroutine.

Error Logging Overview in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*

---

ethchan_config Command

Purpose

Adds adapters to an EtherChannel or removes adapters from an EtherChannel.

Syntax

ethchan_config { **-a** [**-b**] | **-d** } *EtherChannel Adapter*

ethchan_config -c *EtherChannel Attribute NewValue*

ethchan_config -f *EtherChannel*

Description

This command adds adapters to an EtherChannel or removes adapters from an EtherChannel. This command can also be used to modify *EtherChannel* attributes. These additions, deletions or modifications can take place even if the EtherChannel's interface is currently configured; that is, it is not necessary to detach the EtherChannel's interface to add or remove adapters or modify most EtherChannel attributes.

Flags

-a	Adds the specified *Adapter* to the specified *EtherChannel*. If the adapter must be added as a backup adapter, the **-b** flag must be specified.
-b	Specifies that the *Adapter* is being added as a backup adapter. This flag is only valid when used with the **-a** flag.
-c	Changes the specified *Attribute* of the specified *EtherChannel* attribute to the specified *NewValue*.
-d	Deletes the specified *Adapter* from the specified *EtherChannel*. The **-b** flag should not be used with the **-d** flag.
-f	Forces a failover of the specified *EtherChannel*. Note that the failover will only actually occur if the adapter in the idle channel is up: if the adapter in the idle channel is down, the *EtherChannel* will keep operating on the active one and no failover will take place.

Parameters

Adapter	Specifies the adapter to add or delete.
Attribute	Specifies an attribute of the specified EtherChannel.
EtherChannel	Specifies the EtherChannel.
NewValue	Specifies the new value for the specified attribute of the specified EtherChannel.

Exit Status

0	The command completed successfully.
>0	An error occurred.

Examples

1. To add the adapter `ent0` as the backup adapter in the EtherChannel called `ent7`, type:
   ```
   /usr/lib/methods/ethchan_config -a -b ent7 ent0
   ```
2. To change the address to ping attribute of an EtherChannel called `ent7` to `10.10.10.10`, type:
   ```
   /usr/lib/methods/ethchan_config -c ent7 netaddr 10.10.10.10
   ```
3. To force a failover of an EtherChannel called `ent7` from the currently active channel to the idle channel, type:
   ```
   /usr/lib/methods/ethchan_config -f ent7
   ```

Restrictions

The use of the *use_jumbo_frame* attribute cannot be modified by this command. Attempting to do so will print out an error message.

Location

/usr/lib/methods

---

ewallevent Command

Purpose

Broadcasts an event or a rearm event to all users who are logged in.

Syntax

ewallevent [**-c**] [**-h**]

Description

The **ewallevent** script broadcasts a message on an event or a rearm event to all users who are currently logged in to the host when the event or the rearm event occurs. Event or rearm event information is captured and posted by the event response resource manager in environment variables that are generated by the event response resource manager when an event or a rearm event occurs. This script can be used as an action that is run by an event response resource. It can also be used as a template to create other user-defined actions. This script always returns messages in English.

Messages are displayed in this format at the consoles of all users who are logged in when an event or a rearm event occurs for which this script is a response action :

```
Broadcast message from user@host (tty) at hh:mm:ss...

severity event_type occurred for Condition condition_name
on the resource resource_name of resource_class_name at hh:mm:ss mm/dd/yy
The resource was monitored on node_name and resided on {node_names}.
```

Event information is returned about the ERRM environment variables, and also includes the following:

Local Time
Time when the event or rearm event is observed. The actual environment variable supplied by ERRM is ERRM_TIME. This value is localized and converted to readable form before being displayed.

This script captures the environment variable values and uses the **wall** command to write a message to the currently logged-in user consoles.

Flags

–c Instructs **ewallevent** to broadcast the **ERRM_VALUE** of an ERRM event. When the **-c** flag is specified, **ewallevent** broadcasts the SNMP trap message.

–h Writes the script's usage statement to standard output.

Parameters

log_file
Specifies the name of the file where event information is logged. An absolute path for the *log_file* parameter should be specified.

The *log_file* is treated as a circular log and has a fixed size of 64KB. When *log_file* is full, new entries are written over the oldest existing entries.

If *log_file* already exists, event information is appended to it. If *log_file* does not exist, it is created so that event information can be written to it.

Exit Status

0 Script has run successfully.

1 Error occurred when the script was run.

Restrictions

1. This script must be run on the node where the ERRM is running.
2. The **wall** command is used to write a message to currently logged-in user consoles. Refer to the **wall** man page for more information on the **wall** command.

Standard Output

When the **-h** flag is specified, the script's usage statement is written to standard output.

Examples

1. Suppose the **ewallevent** script is a predefined action in the critical-notification response, which is associated with the **/var space used** condition on the resource **/var**. The threshold of the event expression defined for this condition is met, and an event occurs. The critical-notification response takes place, and **ewallevent** is run. The following message is displayed on the consoles of all users who are logged in:

```
Broadcast message from joe@neverland.com (pts/6) at 18:42:03...

Critical event occurred for Condition /var space used
on the resource /var of filesys of IBM.FileSystem at 18:41:50 03/28/02
The resource was monitored on c174n05 and resided on {c174n05}.
```

2. When a rearm event occurs for the **/var space used** condition on the resource **/var**, the following message is displayed on the consoles of all users who are logged in:

```
Broadcast message from joe@neverland.com (pts/6) at 18:42:03...

Critical rearm event occurred for Condition /var space used
on the resource /var of filesys of IBM.FileSystem at 18:41:50 03/28/02
The resource was monitored on c174n05 and resided on {c174n05}.
```

Location

/usr/sbin/rsct/bin/ewallevent

Related Information

Commands: **wall**, **wallevent**

---

ex Command

Purpose

Editor for text files.

Syntax

ex[**-c** *Subcommand*] [**-l**] [**-R**] [**-s**] [**-t***Tag*] [**-V**] [**-w***Number*] [**-v**| **-**] [**+**[*Subcommand*]] [**-r**[*File*]] [*File*...]

Description

The **ex** command starts the ex editor. The ex editor is part of a family of editors that includes the edit editor, which is a simpler version of the ex editor for novice or casual use, and the vi editor, which is a full-screen display editor. Calling the vi editor directly sets environment variables for screen editing. The ex editor is more powerful than a simple line editor because it is a subset of the vi editor and can access the screen editing capabilities of the vi editor.

The *File* parameter specifies the file or files to be edited. If you supply more than one file name, the ex editor edits each file in the specified order.

Notes:

1. To determine how your workstation can perform more efficiently, the ex editor uses the workstation capability database **terminfo** and the type of the workstation you are using from the **TERM** environment variable.
2. The **ex** command affects the current line unless you specify otherwise. In order to work with different parts of the file, you need to know how to address lines in a file.

Flags

-c *Subcommand*	Carries out the ex editor subcommand before editing begins. When a null operand is typed, as in `-c ''` , the editor places the current line at the bottom of the file. (Usually, the ex editor sets the current line at the start of the file or at some specified tag or pattern.)
-l	Indents appropriately for LISP code and accepts the ( ) (open or close parenthesis), { } (left or right brace), and the [[ ]] (double left or double right bracket) characters as text rather than interpreting them as vi subcommands. This flag is active in visual and open modes.
-R	Sets the **readonly** option, preventing you from altering the file.

-s Suppresses all interactive-user feedback. If you use this flag, file input/output errors do not generate a helpful error message. Using this flag is the same as using the **-** flag.

-t *Tag* Loads the file that contains the tag indicated by the parameter *Tag* and positions the editor at that tag. To use this flag, you must first create a database of function names and their locations using the **ctags** command.

-w*Number* Sets the default window size to *Number*.

-v Invokes the vi editor.

> **Note:** When the **-v** flag is selected, an enlarged set of subcommands are available, including screen editing and cursor movement features. See the **vi** command.

-V Invokes the editor in verbose mode.

- Suppresses all interactive-user feedback. If you use this flag, file input/output errors do not generate a helpful error message. Using this flag is the same as using the **-s** flag.

+[*Subcommand*] Begins an edit at the specified editor search or subcommand. When no parameter is typed, the **+***Subcommand* places the current line at the bottom of the file. Usually, the ex editor sets the current line to the start of the file, or to some specified tag or pattern.

-r [*File*] Recovers a file after an editor or system crash. If you do not specify the *File* parameter, a list of all saved files is displayed.

Exit Status

The following exit values are returned:

0 Successful completion.

>0 An error occurred.

Files

/usr/lbin/exrecover	Recover subcommand
/usr/lbin/expreserve	Preserve subcommand
$HOME/.exrc	Editor startup file
./.exrc	Editor startup file
/var/tmp/Ex*nnnnn*	Editor temporary
/var/tmp/Rx*nnnnn*	Names buffer temporary
/var/preserve	Preservation directory

Related Information

The **ctags** command, **ed** command, **edit** command, **vi** command.

---

execerror Command

Purpose

Writes error messages to standard error.

Syntax

execerror

Description

The **execerror** command is executed by an **exec** subroutine when the load of the real program is unsuccessful. It is passed the name of the file being executed and zero or more loader error message strings. Each loader error message string contains an error number followed by error data.

Examples

The **execerror** command is used as follows:

```
char *buffer[1024];
buffer[0] = "execerror" ;
buffer[1] = "name of program that failed to load";
loadquery(L_GETMESSAGES, &buffer[2], sizeof buffer -8);
execvp("/usr/sbin/execerror",buffer);
```

This sample code causes the application to terminate after the messages are written to standard error.

Files

/usr/sbin/execerror	Contains the **execerror** command.

Related Information

The **exec** subroutine, **loadquery** subroutine.

---

execrset Command

Purpose

Runs a program or command attached to an rset.

Syntax

execrset [**-P**] [**-F**] **-c** *CPUlist* [**-m** *MEMlist*] **-e** *command* [*parameters*]

or

execrset [**-P**] [**-F**] [**-S**] *rsetname* [**-e**] *command* [*parameters*]

Description

The **execrset** command executes a command with an attachment to an **rset**. It causes the specified command to be limited to running only on the processors and/or memory regions contained in the rset. An **rset** name in the system registry can be used to specify the processors and/or memory regions the command is allowed to use. Or, an **rset** containing the specified processors and memory regions can be attached to the process.

Flags

-F	Force the **execrset** command to occur. This flag removes a bindprocessor bind and all threads' **rset** in the process before issuing the command. If the **-P** flag is also specified, it detaches the effective **rset** and all threads' **rset** from the process before issuing the command.
-P	Attaches an **rset** as a partition rset.
-c *CPUlist*	List of CPUs to be in the **rset** to be attached to the process which executes the program or command. This can be one or more CPUs or CPU ranges.
-m *MEMlist*	List of memory regions to be in the **rset**. This can be one or more memory regions or ranges.
-e *command* [*parameters*]	Specifies the command to run followed by any parameters. The **-e** flag must be the last flag used in the command.
-S	A hint that indicates that the process must be scheduled to run in single-threaded mode. Only one of the hardware threads of each physical processor that is included in the specified rset will be used to schedule the job. If all the hardware threads of a physical processor are not included in the specified rset, that processor will be ignored. The specified rset must be an exclusive rset or the command fails. Specifying this flag allows jobs to run with single-thread behavior.

Parameters

rsetname The name of the **rset** in the system registry to be attached to the process executing the program or command

Security

The user must have root authority or have **CAP_NUMA_ATTACH** capability. The user must have root authority to attach a partition rset to the command's process (the **-P** flag).

Attention RBAC users and Trusted AIX users: This command can perform privileged operations. Only privileged users can run privileged operations. For more information about authorizations and privileges, see Privileged Command Database in *Security*. For a list of privileges and the authorizations associated with this command, see the **/etc/security/privcmds** database file.

Examples

1. To run the test1 program on CPUs 0-7, type:
   ```
   execrset -c 0-7 -e test1
   ```
2. To run the 'test2 parm1 parm2' program with an attachment to rset named **test/cpus0to15**, type:
   ```
   execrset test/cpus0to15 test parm1 parm2
   ```
3. To run the **ls -l** command on CPU 0, type:
   ```
   execrset -c 0 -e ls -l
   ```

Files

/usr/bin/execrset Contains the **execrset** command.

Related Information

The **attachrset**, **detachrset**, **lsrset**, **mkrset**, and **rmrset** commands.

---

expand Command

Purpose

Writes to standard output with tabs changed to spaces.

Syntax

expand [**-t** *TabList*] [*File ...*]

expand [-*tabstop*]|[-*tab1*,*tab2*,...,*tabn*] [*File* ...]

Description

The **expand** command writes the named files or standard input to standard output, and replaces the tab characters with one or more space characters. Any backspace characters are copied to the output and cause the column position count for tab stop calculations to decrement; the column position count will not decrement below zero.

Note: The *File* parameter must be a text file.

Flags

-t *TabList* Specifies the position of the tab stops. The default value of a tab stop is 8 column positions.

The *TabList* variable must consist of a single positive-decimal integer or multiple positive-decimal integers. The multiple integers must be in ascending order, and must be separated by commas or by blank characters with quotation marks around the integers. The single *TabList* variable sets the tab stops an equal number of column positions apart. The multiple *TabList* variable sets the tab stops at column positions that correspond to the integers in the *TabList* variable.

If the **expand** command processes a tab stop beyond the last one specified in the *TabList* variable, the tab stop is replaced by a single-space character in the output.

Parameters

tabstop Specified as a single argument. It sets *tabstop* SPACE characters apart instead of the default 8.

tab1, tab2,..., tabn Sets TAB characters at the columns specified by -*tab1,tab2,...,tabn*.

Exit Status

This command returns the following exit values:

0 Successful completion.
>0 An error occurred.

Examples

1. To adjust the tab stops an equidistance amount in `text.fil`, enter:

   ```
   expand -t 3 text.fil
   ```

   If `text.fil` contains:

   ```
   1       2        3456789
   ```

   then the **expand** command displays:

   ```
   1  2        3456789
   ```

2. To adjust the tab stops a varied amount in `text.fil`, enter:

   ```
   expand -t 3,15,22  text.fil
   ```

   OR

   ```
   expand -t "3 15 22" text.fil
   ```

   If `text.fil` contains:

   ```
   1      2       3      456789
   ```

   then the **expand** command displays:

   ```
   1  2       3      456789
   ```

Files

/usr/bin/expand Contains the **expand** command.

Related Information

The **newform** command, **tab** command, **unexpand** command, **untab** command.

Files in the *Operating system and device management* introduces you to files and the way you can work with them.

Input and output redirection in the *Operating system and device management* describes how the operating system processes input and output.

---

expfilt Command

Purpose

Exports filter rules to an export file.

Syntax

expfilt [**-p**] [**-q**] [**-r**] [**-v 4** | **6**] **-f** *directory* [**-l** *filt_id_list*]

Description

Use the **expfilt** command to export filter rules into export text files, which can be used by the **impfilt** command. This is useful if you want to define similar rules on multiple machines.

Note: The filter description on one machine might be meaningless or misleading in another machine. This field is not exported.

IPsec filter rules for this command can be configured using the **genfilt** command, IPsec smit (IP version 4 or IP version 6), or Web-based System Manager in the Virtual Private Network submenu.

Flags

-f *directory*	Specifies the directory to create the exported text files. The directory will be created if it does not exist.
-l *filt_id_list*	Lists the IDs of the filter rules you want to export. The filter rule IDs can be separated by ″**,**″ or ″**-**″. If this flag is not used, all the filter rules defined in the filter rule table for the applicable IP versions will be exported.
-p	Allows predefined rules.
-q	Specifies quiet mode. Suppresses output to **stdout**.
-r	Specifies raw mode. Exports filter rules as is and does not reverse direction on rules. Use this flag when filter rules are exported and imported as is; for example, to save a configuration or replicate a configuration to another machine. With the **-r** flag, the direction of the traffic will be preserved. For instance if there is a rule on host 10.0.0.1 to permit inbound traffic from 10.0.0.2, expfilt with the -r flag will write the same filter rule. Omitting the **-r** flag will cause the direction to be switched from inbound to outbound in the export file.
-v	IP version of the filter rules you want to export. The value of **4** specifies IP version 4 and the value of **6** specifies IP version 6. When this flag is not used, both IP version 4 and IP version 6 rules are exported.

Security

Attention RBAC users and Trusted AIX users: This command can perform privileged operations. Only privileged users can run privileged operations. For more information about authorizations and privileges,

see Privileged Command Database in *Security*. For a list of privileges and the authorizations associated with this command, see the **/etc/security/privcmds** database file.

Related Information

The **impfilt** command.

---

explain Command

Purpose

Provides an interactive thesaurus.

Syntax

explain

Description

The **explain** command provides an interactive thesaurus for the English-language phrases found by the **diction** command. Before using the **explain** command, use the **diction** command to obtain a list of poorly worded phrases. When you use the **explain** command, the system prompts you for a phrase and responds with a grammatically acceptable alternative. You can continue typing phrases, or you can exit by entering the Ctrl-D key sequence.

The **explain** command also takes piped input from the command line, as shown in the following example:

```
diction Filename | explain
```

No other command line parameters are valid.

Files

/usr/lib/explain.d	Contains thesaurus.

Related Information

The **diction** command.

---

explore Command

Purpose

Starts the WebExplorer® World Wide Web browser.

Syntax

explore [**-i***FileName*] [**-t***Number*] [**-q**] [[**-url**] *URL*]

Description

The **explore** command opens the WebExplorer main window and connects to the Uniform Resource Locator (URL) for the home document.

Flags

-i*FileName*	Specifies an alternate initialization file, where *FileName* is the full path name of the file to use instead of the default **$HOME/.explore-preferences**. This allows you to start the WebExplorer with an alternate set of user preferences.
-t*Number*	Specifies the number of threads to use for loading images, where *Number* is the number of image loader threads. Each thread is represented in the status area of the main window. A maximum of eight can be specified, and the default is four.
-q	Specifies quiet mode. This suppresses the WebExplorer title window when you start the application and bypasses the confirmation window when you exit.
-url *URL*	Specifies a particular document to load when starting WebExplorer, where *URL* is the URL of the document to load. If WebExplorer has a home document defined, this URL will override it. You do not have to precede the URL with the **-url** flag. If you specify the URL by itself, WebExplorer will accept it.

Exit Status

This command returns the following exit values:

0	Successful completion.
>0	An error occurred.

Security

Access Control: Any User

Auditing Events: N/A

Examples

To start the browser without the title window appearing and go directly to the Dilbert Zone URL, enter:

```
explore -q  http://www.unitedmedia.com/comics/dilbert/
```

or

```
explore -q  -url http://www.unitedmedia.com/comics/dilbert/
```

Files

/usr/lpp/explorer/bin/explore	Contains the **explore** command.
$HOME/.explore-preferences	Contains the initialization file that specifies user preferences for settings such as the number of colors used.
$HOME/.mailcap	Contains the configuration file that maps mimetype to external viewers.
$HOME/.mimetypes	Contains the user-defined configuration file that maps mimetype to external viewers. It is set through the **Configure Viewers** dialog. this file overrides the **.mailcap** settings.

exportfs Command

Purpose

Exports and unexports directories to NFS clients.

Syntax

/usr/sbin/exportfs [**-a**] [**-v**] [**-u**] [**-i**] [**-f***File*] [**-F**] [**-o***Option* [,*Option* ...]] [**-V** *Exported Version*] [*Directory*]

Description

The **exportfs** command makes local directories available for Network File System (NFS) clients to mount. This command is normally invoked during system startup by the **/etc/rc.nfs**file and uses information in the **/etc/exports** file to export one or more directories, which must be specified with full path names.

The **/etc/xtab** file lists directories that are currently exported. To display this file, enter the **exportfs** command without flags or arguments. To alter the file or to alter the characteristics of one of its directories, root users can edit the **/etc/exports** file and run the **exportfs** command. Such alterations can be done at any time. Never edit the **/etc/xtab** file directly.

Notes:

1. You cannot export a directory that is either a parent directory or a subdirectory of one that is currently exported and within the same file system.
2. NFS versions 2 and 3 allow both directories and files to be exported. Only directories can be exported for NFS version 4 access.
3. If two entries for the same directory with different versions 2 (or 3) and 4 exist in the **/etc/exports** file, the **exportfs** command exports both of the entries.
4. If the options for NFS versions 2 (or 3) and 4 are the same for a directory, there can be one entry in the **/etc/exports** file specifying `-vers=3:4`.

Flags

-a	Exports all directories listed in the exports file.
-v	Prints the name of each directory as it is exported or unexported.
-u	Unexports the directories you specify. When used with the **-a** flag, unexports all exported directories. When used with both the **-a** and **-f** flags, unexports all directories in the specified export file.
-i	Allows the exporting of directories not specified in the exports file or ignores the options in the exports file. Unless the **-f** flag is used to specify an alternate file, the **exportfs** command will normally consult the **/etc/exports** file for the options associated with the exported directory.″
-f *File*	Specifies an export file, instead of the **/etc/exports** file, that contains a list of directories that you can export. This file should follow the same format as the **/etc/exports** file. NOTE: This alternate file will not be used for exporting directories automatically when the system and NFS is started. The **/etc/exports** file is the only file that is supported for specifying directories to export at system start.
-F	Specifies that a forced unexport should be performed. Use this flag only with the **-u** flag. This flag has no effect when unexporting a V2/V3 export. A V4 unexport can fail due to associated state. This flag forces the release of any state associated with a V4 export.

-o*Options*

Specifies the optional characteristics for the directory being exported. You can enter more than one variable by separating them with commas. For options taking a *Client* parameter, *Client* can specify a hostname, a dotted IP address, a network name, or a subnet designator. A subnet designator is of the form *"@host/mask"*, where *host* is either a hostname or a dotted IP address and *mask* specifies the number of bits to use when checking access. If *mask* is not specified, a full mask is used. For example, the designator `@client.group.company.com/16` will match all Clients on the `company.com` subnet. A designator of `@client.group.company.com/24` will match only the Clients on the `group.company.com` subnet. Choose from the following options:

ro Exports the directory with read-only permission. If not specified, the directory is exported with read-write permission.

ro=*Client[:Client]*
Exports the directory with read-only permission to the specified Clients. Exports the directory with read-write permissions to Clients not specified in the list. A read-only list cannot be specified if a read-write list has been specified.

rw Exports the directory with read-write permission to all Clients.

rw=*Client [:Client]*
Exports the directory with read-write permission to the specified Clients. Exports the directory read-only to Clients not in the list. A read-write list cannot be specified if a read-only list has been specified.

anon =*UID*
Uses the *UID* value as the effective user ID, if a request comes from a root user.

The default value for this option is -2. In NFS version 2 and NFS version 3, setting the value of the *anon* option to -1 disables anonymous access. Thus, by default, secure NFS accepts nonsecure requests as anonymous, and users who want more security can disable this feature by setting *anon* to a value of -1.

root=*Client[:Client]*
Allows root access from the specified clients in the list. Putting a host in the root list does not override the semantics of the other options. For example, this option denies the mount access from a host present in the root list but absent in the access list.

access=*Client[:Client,...]*
Gives mount access to each client listed. A client can be either a host name or a net group name. Each client in the list is first checked for in the **/etc/netgroup** database and then in the **/etc/hosts** database. The default value allows any machine to mount the given directory.

secure Requires clients to use a more secure protocol when accessing the directory.

-o *Options* (continued)

sec=*flavor[:flavor...]*

This option is used to specify a list of security methods that may be used to access files under the exported directory. Most exportfs options can be clustered using the **sec** option. Options following a **sec** option are presumed to belong with the preceding **sec** option. Any number of **sec** stanzas may be specified, but each security method can be specified only once. Within each **sec** stanza the **ro**, **rw**, **root**, and **access** options may be specified once. Only the **public**, **anon** and **vers** options are considered global for the export. If the **sec** option is used to specify any security method, it must be used to specify all security methods. In the absence of any **sec** option, all authentication flavors are allowed.

Allowable flavor values are:

sys UNIX authentication. This is the default method.

dh DES authentication.

none Allow mount requests to proceed with anonymous credentials if the mount request uses an authentication flavor not specified in the export.

krb5 Kerberos. Authentication only.

krb5i Kerberos. Authentication and integrity.

krb5p Kerberos. Authentication, integrity, and privacy.

The **secure** option may be specified, but not in conjunction with a **sec** option. The **secure** option is deprecated and may be eliminated. Use **sec=dh** instead.

vers=*version_number[:version_number...]*

Specifies which versions of NFS are allowed to access the exported directory. Valid versions are 2, 3, and 4. Versions 2 and 3 cannot be selected exclusively. Specifying either version 2 or version 3 will allow access by both NFS version 2 and NFS version 3. Version 4 can be selected exclusively. The default is to allow access using NFS protocol versions 2 and 3.

exname=*external-name*

Exports the directory by the specified external name. The external name must begin with the **nfsroot** name. See the description of the **/etc/exports** file for a description of the **nfsroot** name. This option applies only to directories exported for access by NFS version 4 protocol.

deleg={yes | no}

Enables or disables file delegation for the specified export. This option overrides the system-wide delegation enablement for this export. The system-wide enablement is done through **nfso**.

-o *Options* (continued)

refer=*rootpath@host***[+***host***][:***rootpath@host***[+***host***]]**
A namespace referral will be created at the specified path. The referral directs clients to the specified alternate locations where they can continue operations. A referral is a special object. If a nonreferral object exists at the specified path, the export is disallowed and an error message is printed. If nothing exists at the specified path, a referral object is created there that includes the path name directories leading to the object. Multiple referrals can be created within a file system. A referral cannot be specified for the **nfsroot**. The name `localhost` cannot be used as a *hostname*. This **refer** option is allowed only for version 4 exports. If the export specification allows version 2 or version 3 access, an error message will be printed and the export will be disallowed. Unexporting the referral object has the effect of removing the referral locations information from the referral object. The object itself is not removed by unexporting. Use **rm** if you want to remove the object. The administrator must ensure that appropriate data is available at the referral servers. This option is available only on AIX 5L Version 5.3 with the 5300-03 Recommended Maintenance package or later.
Note: A referral export can only be made if replication is enabled on the server. Use **chnfs -R on** to enable replication.

-o *Options* (continued)

replicas=*rootpath@host***[+***host***][:***rootpath@host***[+***host***]]**

Replica location information will be associated with the export path. The replica information can be used by NFS version 4 clients to redirect operations to the specified alternate locations if the current server becomes unavailable. The administrator should ensure that appropriate data is available at the replica servers. Because replica information applies to an entire file system, the specified path must be the root of a file system. If the path is not a file system root, the export is disallowed and an error message is printed. The name `localhost` cannot be used as a *hostname*. This **replicas** option is meaningful only for version 4 exports. If the option is used on an export that allows version 2 or version 3 access, the operation is allowed, but the replica information is ignored by the version 2 and version 3 servers. If the directory being exported is not in the replica list, the entry *exported directory@current host* will be added as the first replica location. This option is available only on AIX 5.3 with 5300-03 or later.

A replica export can only be made if replication is enabled on the server. By default, replication is not enabled. If replica exports will be made at system boot, replication should be enabled by using the **chnfs -R on** command. Replica locations can also be specified for the **nfsroot**. This can be done only using **chnfs -R** *host*[+*host*]. If the current host is not specified in the list, it will be added as the first replica host. The *rootpath* is not needed or allowed in this case because **nfsroot** is replicated only to the **nfsroots** of the specified hosts. The **chnfs** program can be used to enable or disable replication. Changing the replication mode can only be done if no NFS version 4 exports are active. If the server's replication mode is changed, file handles issued by the server during the previous replication mode will not be honored by the server. This can cause application errors on clients holding old file handles. Be careful when changing the replication mode of the server. If possible, all clients who have mounts to the server should unmount them before the server's replication mode is changed. The replica location information associated with the directory can be changed by modifying the replica list and reexporting the directory. The new replica information replaces the old replica information.

NFS clients are expected to refresh replica information on a regular basis. If the server changes the replica information for an export, it might take time for the client to notice. This is not much of a problem if new replica locations are added, because clients holding the old information still have correct, if incomplete, replica information. Removing replica information can be problematic because it can result in clients holding incorrect replica information for a period of time. To aid clients in detecting the new information, **exportfs** will attempt to touch the replicated directory. This changes the timestamps on the directory, which in turn causes the client to refetch the directory's attributes. This operation might not be possible, however, if the replicated file system is read-only. When changing replica information for a directory, be aware that there could be some latency between changing the information and clients noticing the new information.

-o *Options* (continued)

noauto Accepts the replicas specification as-is. Does not automatically insert the primary hostname as one of the replica locations if it has not been specified.

scatter Defines how the alternate locations list is generated from the servers specified on the **refer** or **replicas** option. If the **noauto** option is not used, the alternate locations list also includes the primary host name as one of the replica locations. The **scatter** option applies only to directories exported for access by NFS version 4 protocol. The **scatter** option has three allowable values:

- **full** All of the servers are scattered to form the combinations of alternate locations.
- **partial** The first location of all the combinations is fixed to the first server specified on the **refer** or **replicas** option. The rest of the locations and the first location are scattered as if they are scattered using the `scatter=full` method.
- **none** No scatter is to be used. The value can also be used to disable scattering if enabled previously.

Whenever the attributes of a Client change, all export entries that contain that Client as a parameter should be exported again. Events that can change a Client's attributes include modifying a netgroup or changing the IP address of a client. Failure to do so can result in the server using old client information.

-V *Exported Version* Specifies the version number. Valid version numbers are 2, 3 and 4.

Solaris Compatibility

The **exportfs** command may be invoked as **share**, **shareall**, **unshare**, or **unshareall**. When the **exportfs** command is invoked as **share** or **shareall**, the functionality is equivalent to **exportfs** and **exportfs -a**, respectively, except that the **sec** option must be used to specify the security methods. When the **exportfs** command is invoked as **unshare** or **unshareall**, the functionality is equivalent to **exportfs -u** and **exportfs -u -a**, respectively.

Security

Attention RBAC users and Trusted AIX users: This command can perform privileged operations. Only privileged users can run privileged operations. For more information about authorizations and privileges, see Privileged Command Database in *Security*. For a list of privileges and the authorizations associated with this command, see the **/etc/security/privcmds** database file.

Examples

1. To export all directories in the **/etc/exports** file, enter:
   ```
   exportfs -a
   ```
2. To export one directory from the **/etc/exports** file, enter:
   ```
   exportfs /home/notes
   ```
   In this example, the `/home/notes` directory is exported.

   Note: For this command to work, the **/home/notes** directory must be specified in the **/etc/exports** file.
3. To unexport a directory, enter:
   ```
   exportfs -u /home/notes
   ```
   In this example, the `/home/notes` directory is unexported.
4. To display the name of the directory currently being exported, enter:
   ```
   exportfs -v
   ```
5. To export a directory that is not specified in the **/etc/exports** file, enter:

```
exportfs -i /home/zeus
```

In this example, the `/home/zeus` directory is exported without restrictions.

6. To export a directory and give netgroup members permission to access this directory, enter:

   ```
   exportfs access=cowboys:oilers /home/notes -o
   ```

   In this example, the `/home/notes` directory is exported and permits users of `cowboys` and `oilers` host machines to have access.

7. To export a directory with different options from the **/etc/exports** file, enter:

   ```
   exportfs -i -o -root=zorro:silver /directory
   ```

   In this example, the `/directory` directory is exported and allows root user access to `zorro` and `silver` host machines, regardless of the access permissions specified in the **/etc/exports** file.

8. To export the **/common/docs** directory with write permissions to clients using Kerberos authentication, but read-only permissions to clients using UNIX authentication, add the following text to the **/etc/exports** file:

   ```
   /common/docs -sec=krb5,rw,sec=sys,ro
   ```

   Then enter `exportfs /common/docs` to perform the export.

9. To create a referral at **/usr/info** to the **/usr/info** directory on host `infoserver`, add the following line to **/etc/exports** and then export **/usr/info**:

   ```
   /usr/info -vers=4,refer=/usr/info@infoserver
   ```

10. To specify replicas for the **/common/info** directory at hosts `backup1` and `backup2`, add the following line to **/etc/exports** and then export **/common/info**:

    ```
    /common/info -vers=4,replicas=/common/info@backup1:/common/info@backup2,<other options>
    ```

11. To export the **/common/docs** directory with both version 3 and version 4, enter the following command:

    ```
    exportfs -V 3:4 /common/docs
    ```

12. To export all of the version 4 entries in the **/etc/exports** file, enter the following command:

    ```
    exportfs -a -V 4
    ```

13. To unexport the **/common/docs** directory only for version 3, enter the following command:

    ```
    exportfs -u -V 3 /common/docs
    ```

14. To unexport all of the version 3 entries in the **/etc/xtab** file, enter the following command:

    ```
    exportfs -ua -V 3
    ```

15. To specify referrals for the **/common/docs** directory at hosts named `s1`, `s2` and `s3` and scatter them fully, add the following line to the **/etc/exports** file and then export the **/common/docs** directory:

    ```
    /common/docs -vers=4,refer=/common/docs@s1:/common/docs@s2:/common/docs@s3,scatter=full
    ```

16. To specify replicas for the **/common/docs** directory at hosts named `s1`, `s2`, `s3` and `s4` and scatter them partially (the first fail over server is `s1` for all combinations), add the following line to the **/etc/exports** file and then export the **/common/docs** directory:

    ```
    /common/docs -vers=4,noauto,replicas=/common/docs@s1:/common/docs@s2:/common/docs@s3:/common/docs@s4,
    scatter=partial
    ```

Files

/etc/exports	Lists the directories that the server can export.
/etc/xtab	Lists currently exported directories.
/etc/hosts	Contains an entry for each host on the network.
/etc/netgroup	Contains information about each user group on the network.
/etc/rc.nfs	Contains the startup script for the NFS and NIS daemons.

Related Information

The **chnfsexp** command, **mknfsexp** command, **rmnfsexp** command, **showmount** command.

How to Export a File System Using Secure NFS in . *Security*

List of NFS commands, List of NFS files, and Network File System in *Networks and communication management*.

---

exportvg Command

Purpose

Exports the definition of a volume group from a set of physical volumes.

Syntax

exportvg *VolumeGroup*

Description

The **exportvg** command removes the definition of the volume group specified by the *VolumeGroup* parameter from the system. Since all system knowledge of the volume group and its contents are removed, an exported volume group can no longer be accessed. The **exportvg** command does not modify any user data in the volume group.

A volume group is a nonshared resource within the system; it should not be accessed by another processor until it has been explicitly exported from its current processor and imported on another. The primary use of the **exportvg** command, coupled with the **importvg** command, is to allow portable volumes to be exchanged between processors. Only a complete volume group can be exported, not individual physical volumes.

Using the **exportvg** command and the **importvg** command, you can also switch ownership of data on physical volumes shared between two processors.

> **Note:** To use this command, you must either have root user authority or be a member of the **system** group.

You can use the Volumes application in Web-based System Manager (wsm) to change volume characteristics.

You can use the Web-based System Manager Volumes application (**wsm lvm** fast path) to run this command. You could also use the System Management Interface Tool (SMIT) **smit exportvg** fast path to run this command.

> **Notes:**
>
> 1. A volume group that has a paging space volume on it cannot be exported while the paging space is active. Before exporting a volume group with an active paging space volume, ensure that the paging space is not activated automatically at system initialization, and then reboot the system.
> 2. The mount point information of a logical volume would be missing from the LVCB (logical volume control block) if it is longer than 128 characters. Please make a note of the mount points that are longer than 128 characters as you will need to edit the **/etc/filesystems** file manually upon executing **importvg** command to import this volume group completely.

Exit Status

This command returns the following exit values:

0	Successful completion.
>**0**	An error occurred.

Security

Attention RBAC users and Trusted AIX users: This command can perform privileged operations. Only privileged users can run privileged operations. For more information about authorizations and privileges, see Privileged Command Database in *Security*. For a list of privileges and the authorizations associated with this command, see the **/etc/security/privcmds** database file.

Examples

To remove volume group `vg02` from the system, enter:

```
exportvg vg02
```

Note: The volume group must be varied off before exporting.

The definition of `vg02` is removed from the system and the volume group cannot be accessed.

Files

/usr/sbin	Directory where the **exportvg** command resides.

Related Information

The **importvg** command, **varyoffvg** command, **varyonvg** command.

The Logical volume storage in *Operating system and device management* explains the Logical Volume Manager, physical volumes, logical volumes, volume groups, organization, ensuring data integrity, and allocation characteristics.

For information on installing the Web-based System Manager, see Chapter 2: Installation and System Requirements in *AIX Version 6.1 Web-based System Manager Administration Guide*.

The System management interface tool in *Operating system and device management* explains the structure, main menus, and tasks that are done with SMIT.

---

expr Command

Purpose

Evaluates arguments as expressions.

Syntax

expr *Expression*

Description

The **expr** command reads the *Expression* parameter, evaluates it, and writes the result to standard output.

You must apply the following rules to the *Expression* parameter:

- Separate each term with blanks.

- Precede characters special to the shell with a \ (backslash).
- Quote strings containing blanks or other special characters.

Integers may be preceded by a unary hyphen. Internally, integers are treated as 32-bit, twos complement numbers.

Note: The **expr** command returns 0 to indicate a zero value, rather than the null string.

The following items describe *Expression* parameter operators and keywords. Characters that need to be escaped are preceded by a \ (backslash). The items are listed in order of increasing precedence, with equal precedence operators grouped within { } (braces):

Expression1 **|** *Expression2*	Returns *Expression1* if it is neither a null value nor a 0 value; otherwise, it returns *Expression2*.
Expression1 **\&** *Expression2*	Returns *Expression1* if both expressions are neither a null value nor a 0 value; otherwise, it returns a value of 0.
Expression1 **{ =, \>, \>=, \<, \<=, != }** *Expression2*	Returns the result of an integer comparison if both expressions are integers; otherwise, it returns the result of a string comparison.
Expression1 **{+, -}** *Expression2*	Adds or subtracts integer-valued arguments.
Expression1 **{ *, /, % }** *Expression2*	Multiplies, divides, or provides the remainder from the division of integer-valued arguments.
Expression1 **:** *Expression2*	Compares the string resulting from the evaluation of *Expression1* with the regular expression pattern resulting from the evaluation of *Expression2*. Regular expression syntax is the same as that of the **ed** command, except that all patterns are anchored to the beginning of the string (that is, only sequences starting at the first character of a string are matched by the regular expression). Therefore, a ^ (caret) is not a special character in this context. Normally the matching operator returns the number of characters matched (0 on failure). If the pattern contains a subexpression, that is: `\( Expression \)` then a string containing the actual matched characters is returned. A collating sequence can define equivalence classes for use in character ranges. See ″Understanding Locale Environment Variables″ in *AIX Version 6.1 National Language Support Guide and Reference* for more information on collating sequences and equivalence classes.

Note: The following string arguments are extensions beyond that of the standards, and the behavior may be different across operating systems. These string arguments are NOT portable.

match *String1 String2*	Same as *Expression1* **:** *Expression2*.
length *String1*	Returns the length of the *String1*.
index *String1 String2*	Returns the first position in *String1* where any character in *String2* exists.
substr *String1 StartPosition Length*	

Returns a string that starts with the character at *StartPosition* in *String1* and continuies for *Length* characters

Exit Status

This command returns the following exit values:

0 The *Expression* parameter evaluates to neither null nor 0.
1 The *Expression* parameter evaluates to null or 0.
2 The *Expression* parameter is not valid.
>2 An error occurred.

Note: After parameter processing by the shell, the **expr** command cannot distinguish between an operator and an operand except by the value. Thus, if the value of `$a` is `j`, the command:

```
expr $a = j
```

looks like:

```
expr j = j
```

after the shell passes the arguments to the **expr** command. The following is also true:

```
expr X$a = Xj
```

Examples

1. To modify a shell variable, enter:

   ```
   COUNT=`expr $COUNT + 1`
   ```

   This adds `1` to the shell variable `$COUNT`. The **expr** command is enclosed in grave accents, which causes the shell to substitute the standard output from the **expr** command into the `COUNT=` command. The `$COUNT` variable must be initialized before using.

2. To find the length of the **$STR** shell variable, enter:

   ```
   LENGTH=`expr $STR : ".*"`
   ```

   This sets the `LENGTH` variable to the value given by the: (colon) operator. The pattern .* (dot, asterisk) matches any string from beginning to end, so the colon operator gives the length of the `$STR` variable as the number of characters matched. Note that `.*` must be within quotes to prevent the shell from treating the * (asterisk) as a pattern-matching character. The quotes are not part of the pattern.

   If the `$STR` variable is set to the null string or contains any white space (blanks or tabs), then the command displays the error message `expr: syntax error`. This happens because the shell does not normally pass null strings to commands. In this case, the **expr** command sees only:

   ```
   :.*
   ```

   The shell also removes the single quotation marks. This does not work because the colon operator requires two values. The problem is fixed by enclosing the shell variable in double quotation marks:

   ```
   LENGTH=`expr "$STR" : ".*"`
   ```

   Now if the value of the `$STR` variable is null, the `LENGTH` variable is set to a value of 0. Enclosing shell variables in double quotation marks is generally recommended. Do not enclose shell variables in single quotation marks.

3. To use part of a string, enter:

   ```
   FLAG=`expr "$FLAG" : "-*\(.*\)"`
   ```

This removes leading hyphens, if any, from the $FLAG shell variable. The colon operator gives the part of the FLAG variable matched by the subexpression enclosed between \(and \) characters (backslash, open parenthesis and backslash, close parenthesis). If you omit the \(and \) subexpression characters, the colon operator gives the number of characters matched.

If the $FLAG variable is set to - (hyphen), the command displays a syntax error message. This happens because the shell substitutes the value of the $FLAG variable before running the **expr** command. The **expr** command does not know that the hyphen is the value of a variable. It can only see:

```
- : -*\(.*\)
```

and it interprets the first hyphen as the subtraction operator. To eliminate this problem, use:

```
FLAG=`expr "x$FLAG" : "x-*\(.*\)"`
```

4. To use the **expr** command in an **if** statement, enter:

```
if expr "$ANSWER" : "[yY]" >/dev/null
then
echo ANSWER begins with "y" or "Y"
fi
```

If the $ANSWER variable begins with y or Y, the then part of the **if** statement is performed. If the match succeeds, the result of the expression is 1 and the **expr** command returns an exit value of 0, which is recognized as the logical value True by the **if** statement. If the match fails, the result is 0 and the exit value 1 (False).

Redirecting the standard output of the **expr** command to the **/dev/null** special file discards the result of the expression. If you do not redirect it, the result is written to the standard output, which is usually your workstation display.

5. Consider the following expression:

```
expr "$STR" = "="
```

If the $STR variable has the value = (equal sign), then after the shell processes this command the **expr** command sees the expression:

```
= = =
```

The **expr** command interprets this as three = operators in a row and displays a syntax error message. This happens whenever the value of a shell variable is the same as that of one of the **expr** operators. You can avoid this problem by phrasing the expression as:

```
expr "x$STR" = "x="
```

6. To return the length of the $SHELL environment variable, /usr/bin/ksh, enter:

```
expr length $SHELL
```

The following is displayed:

```
12
```

7. To return the first position of where any characters in the string ″de″ is found in ″abcdef″, enter:

```
expr index abcdef de
```

The following is displayed:

```
4
```

8. To return the first position of where any characters in the string ″fd″ is found in ″abcdef″, enter:

```
expr index abcdef fd
```

The following is displayed:

```
4
```

9. To return the string starting at position 11, for a length of 6 of the string ″Goodnight Ladies″, enter:

```
expr substr "Goodnight Ladies" 11 6
```

The following is displayed:

```
Ladies
```

Files

/usr/bin/expr	Contains the **expr** command.

Related Information

The **bsh** command, **csh** command, **ed** command, **ksh** command.

Commands in *Operating system and device management*.

National Language Support Overview in *AIX Version 6.1 National Language Support Guide and Reference*.

---

exptun Command

Purpose

Exports a tunnel definition and, optionally, all the user defined filter rules associated with the tunnel. Creates a tunnel export file and an optional filter rule export file that can be used for the tunnel partner.

Syntax

exptun [**-v 4**|**6**] **-f** *directory* [**-t** *tid_list*] [**-r**] [**-l manual**]

Description

Use the **exptun** command to create a tunnel context export file and, optionally, a filter rule appendage file for a tunnel partner to import. This command does not activate a tunnel, it simply creates the required files for the tunnel partner.

> **Notes:** Generated export files contain keys used by the tunnel. Protect these files with the operating system file system protection features.

Flags

-f	Defines the directory where the export files are to be written. The directory will be created if it does not exist. The export files may then be sent to the tunnel partner to be imported. It is recommended that export files for each tunnel partner have a different directory specification.
-l	The type of the tunnel(s) you want to export. If manual is specified, only manual ibm tunnel(s)are exported.
-r	Exports all the user defined filter rules associated with the tunnel(s). If this flag is not used, only the tunnel definitions will be exported.
-t	Specifies the list of tunnel IDs to be used for the export files. The list may be specified as a sequence of tunnel IDs separated by a ″**,**″ or ″**-**″ (1, 3, 10, 50-55). If this flag is not used, all tunnel definitions from the tunnel database will be exported.
-v	The IP version of the tunnels being exported. Value **4** specifies IP version 4 tunnels. Value **6** specifies IP version 6 tunnels. If this flag is not used, both IP version 4 and IP version 6 tunnel definitions will be exported.

Security

Attention RBAC users and Trusted AIX users: This command can perform privileged operations. Only privileged users can run privileged operations. For more information about authorizations and privileges,

see Privileged Command Database in *Security*. For a list of privileges and the authorizations associated with this command, see the **/etc/security/privcmds** database file.

Related Information

The **chtun** command, **gentun** command, **imptun** command, **lstun** command, **mktun** command, and **rmtun** command.

---

extendlv Command

Purpose

Increases the size of a logical volume by adding unallocated physical partitions from within the volume group.

Syntax

To Add Available Physical Partitions

extendlv [**-a** *Position*] [**-e** *Range*] [**-u** *Upperbound*] [**-s** *Strict*] *LogicalVolume Partitions* [*PhysicalVolume ...*]

To Add Specific Physical Partitions

extendlv [**-m***MapFile*] *LogicalVolume Partitions*

Description

The **extendlv** command increases the number of logical partitions allocated to the *LogicalVolume* by allocating the number of additional logical partitions represented by the *Partitions* parameter. The *LogicalVolume* parameter can be a logical volume name or a logical volume ID. To limit the allocation to specific physical volumes, use the names of one or more physical volumes in the *PhysicalVolume* parameter; otherwise, all the physical volumes in a volume group are available for allocating new physical partitions.

By default, the logical volume is expanded using the existing characteristics that are displayed when you use the **lslv** command. To override these existing characteristics for the new partitions only, choose different values for these characteristics by using the flags.

The default maximum number of partitions for a logical volume is 512. Before extending a logical volume more than 512 logical partitions, use the **chlv** command to increase the default value.

The default allocation policy is to use a minimum number of physical volumes per logical volume copy, to place the physical partitions belonging to a copy as contiguously as possible, and then to place the physical partitions in the desired region specified by the **-a** flag. Also, by default, each copy of a logical partition is placed on a separate physical volume.

You can specify logical volumes sizes in 512 Blocks/KB/MB/GB when using the **extendlv** command. (See “Examples” on page 430.)

Notes:

1. When extending a striped logical volume, the number of partitions must be in an even multiple of the striping width.
2. It is recommended that a logical volume using a large number of partitions (more than 800MB) be extended gradually in sections.
3. Changes made to the logical volume are not reflected in the file systems. To change file system characteristics, use the **chfs** command.
4. You must either have root user authority or be a member of the **system** group to use this command.

5. The **extendlv** command is not allowed on a snapshot volume group.

You can use the Volumes application in Web-based System Manager to change volume characteristics. You could also use the System Management Interface Tool (SMIT) **smit extendlv** fast path to run this command.

Flags

Note: The **-e** and **-s** flags are not valid with a striped logical volume.

-a *Position*
Sets the intraphysical volume allocation policy (the position of the logical partitions on the physical volume). The *Position* variable can be one of the following:

m Allocates logical partitions in the outer middle section of each physical volume. This is the default position.

c Allocates logical partitions in the center section of each physical volume.

e Allocates logical partitions in the outer edge section of each physical volume.

ie Allocates logical partitions in the inner edge section of each physical volume.

im Allocates logical partitions in the inner middle section of each physical volume.

-e *Range*
Sets the interphysical volume allocation policy (the number of physical volumes to extend across, using the volumes that provide the best allocation). The value of the *Range* variable is limited by the *Upperbound* variable (set with the -u flag) and can be one of the following:

x Allocates logical partitions across the maximum number of physical volumes.

m Allocates logical partitions across the minimum number of physical volumes.

-m *MapFile*
Specifies the exact physical partitions to allocate. Partitions are used in the order given by the file designated by the *MapFile* parameter. All physical partitions belonging to a copy are allocated before allocating for the next copy. The *MapFile* format is:

PVname:PPnum1[-PPnum2]
where *PVname* is a physical volume name (for example, `hdisk0`). It is one record per physical partition or a range of consecutive physical partitions.

PVname Name of the physical volume as specified by the system.

PPnum Physical partition number.

Important: When you use map files, you must understand and adhere to all LV-allocation parameters such as strictness, upperbound, and stripe width. Using map files bypasses the checks done in the LVM-allocation routines. This is important for striped LVs, which are assumed to have a typical striped allocation pattern conforming to the stripe width.

-s *Strict*
Determines the strict allocation policy. Copies of a logical partition can be allocated to share or not to share the same physical volume. The *Strict* variable is represented by one of the following:

y Sets a strict allocation policy, so copies for a logical partition cannot share the same physical volume.

n Does not set a strict allocation policy, so copies for a logical partition can share the same physical volume.

s Sets a super strict allocation policy, so that the partitions allocated for one mirror cannot share a physical volume with the partitions from another mirror.

Note: When changing a non superstrict logical volume to a superstrict logical volume you must specify physical volumes or use the **-u** flag.

-u *Upperbound* Sets the maximum number of physical volumes for new allocation. The value of the *Upperbound* variable should be between one and the total number of physical volumes. When using super strictness, the upper bound indicates the maximum number of physical volumes allowed for each mirror copy. When using striped logical volumes, the upper bound must be multiple of *Stripe_width*.

Security

Attention RBAC users and Trusted AIX users: This command can perform privileged operations. Only privileged users can run privileged operations. For more information about authorizations and privileges, see Privileged Command Database in *Security*. For a list of privileges and the authorizations associated with this command, see the **/etc/security/privcmds** database file.

Examples

1. To increase the size of the logical volume represented by the `lv05` directory by three logical partitions, type:

   ```
   extendlv lv05 3
   ```

2. To request a logical volume named `lv05` with a minimum size of 10MB, type:

   ```
   extendlv lv05 10M #
   ```

   The **extendlv** command will determine the number of partitions needed to create a logical volume of at least that size.

   You can use uppercase and lowercase letters as follows:

   ```
   B/b       512 byte blocks
   K/k       KB
   M/m       MB
   G/g       GB
   ```

Files

/usr/sbin/ Directory where the **extendlv** command resides.

Related Information

The **chfs** command, **chlv** command, **chpv** command, **lslv** command, **mklv** command, **mklvcopy** command.

The Logical volume storage in *Operating system and device management* explains the Logical Volume Manager, physical volumes, logical volumes, volume groups, organization, ensuring data integrity, and allocation characteristics.

For information on installing the Web-based System Manager, see Chapter 2: Installation and System Requirements in *AIX Version 6.1 Web-based System Manager Administration Guide*.

The System management interface tool in *Operating system and device management* explains the structure, main menus, and tasks that are done with SMIT.

---

extendvg Command

Purpose

Adds physical volumes to a volume group.

Syntax

extendvg [**-f**] *VolumeGroup PhysicalVolume ...*

Description

The **extendvg** command increases the size of the *VolumeGroup* by adding one or more *PhysicalVolumes*.

The physical volume is checked to verify that it is not already in another volume group. If the system believes the physical volume belongs to a volume group that is varied on, it exits. But if the system detects a description area from a volume group that is not varied on, it prompts the user for confirmation in continuing with the command. The previous contents of the physical volume are lost, so the user must be cautious when using the override function.

Note: To use this command, you must either have root user authority or be a member of the **system** group.

For volume groups created prior to AIX 5.3, or for volume groups created on AIX 5.3 but varied on with the **varyonvg -M** flag, the **extendvg** will fail if the physical volume has a max transfer size that is smaller than the logical track group size of the volume group. For volume groups created on AIX 5.3 and varied on without the **varyonvg -M** flag, extendvg will dynamically lower the logical track group size of the volume group if the physical volume has a max transfer size that is smaller than the logical track group size of the volume group.

Note: The **extendvg** command is not allowed on a snapshot volume group.

You can use the Volumes application in Web-based System Manager (wsm) to change volume characteristics. You could also use the System Management Interface Tool (SMIT) **smit extendvg** fast path to run this command.

Note: This command will fail to add a disk to the volume group if the disk indicates that it is managed by a third party volume manager. To override and clear the disk of the third party volume manger use **chpv -C** *HDiskName*.

Note: When extending a concurrent Volume Group (VG), you must first ensure that each new disk to be added to the VG has a Physical Volume Identifier (PVID) assigned, and that the PVID stored in the Object Data Manager (ODM) is the same one on every node. When using the Cluster Single Point of Control (C-SPOC) utility to extend the VG, this check is done automatically.

Flags

-f Forces the physical volume to be added to the specified volume group unless it is a member of another volume group in the Device Configuration Database or of a volume group that is active.

Security

Attention RBAC users and Trusted AIX users: This command can perform privileged operations. Only privileged users can run privileged operations. For more information about authorizations and privileges, see Privileged Command Database in *Security*. For a list of privileges and the authorizations associated with this command, see the **/etc/security/privcmds** database file.

Examples

To add physical volumes `hdisk3` and `hdisk8` to volume group `vg3`, enter:

```
extendvg vg3 hdisk3 hdisk8
```

Note: The volume group must be varied on before extending.

Restrictions

The **extendvg** command cannot be run on a snapshot volume group.

Files

/usr/sbin/extendvg	Contains the **extendvg** command.

Related Information

The **reducevg** command.

The Logical volume storage in *Operating system and device management* explains the Logical Volume Manager, physical volumes, logical volumes, volume groups, organization, ensuring data integrity, and allocation characteristics.

For information on installing the Web-based System Manager, see Chapter 2: Installation and System Requirements in *AIX Version 6.1 Web-based System Manager Administration Guide*.

The System management interface tool in *Operating system and device management* explains the structure, main menus, and tasks that are done with SMIT.

---

f Command

Purpose

Shows user information. This command is the same as the **finger** command.

Syntax

{ **f** | **finger** } [[**-b**] [**-h**] [**-l**] [**-p**]] | [**-i**] [**-q**] [**-s**] [**-w**]]

[**-f**] [**-m**] [*User* | *User* **@***Host* | **@***Host*]

Description

The **/usr/bin/f** command displays information about the users currently logged in to a host. The format of the output varies with the options for the information presented.

Default Format

The default format includes the following items:

- Login name
- Full user name
- Terminal name
- Write status (an * (asterisk) before the terminal name indicates that write permission is denied)

For each user on the host, the default information list also includes, if known, the following items:

- Idle time (Idle time is minutes if it is a single integer, hours and minutes if a : (colon) is present, or days and hours if a ″d″ is present.)
- Login time
- Site-specific information

  The site-specific information is retrieved from the gecos field in the **/etc/passwd** file. The gecos field may contain the Full user name followed by a comma. All information that follows the comma is displayed by the finger command with the Site-specific information.

Longer Format

A longer format is used by the **f** command whenever a list of user's names is given. (Account names as well as first and last names of users are accepted.) This format is multiline, and includes all the information described above along with the following:

- User's **$HOME** directory
- User's login shell
- Contents of the **.plan** file in the user's **$HOME** directory
- Contents of the **.project** file in the user's **$HOME** directory

The **f** command may also be used to look up users on a remote system. The format is to specify the user as *User@Host*. If you omit the user name, the **f** command provides the standard format listing on the remote system.

Create the **.plan** and **.project** files using your favorite text editor and place the files in your **$HOME** directory. The **f** command uses the **toascii** subroutine to convert characters outside the normal ASCII character range when displaying the contents of the **.plan** and **.project** files. The **f** command displays a `M-` before each converted character.

When you specify users with the *User* parameter, you can specify either the user's first name, last name, or account name. When you specify users, the **f** command, at the specified host, returns information about those users only in long format.

For other information about the **f** command, see ″Installation of TCP/IP″ in *Networks and communication management*.

Flags

-b Gives a brief, long-form listing.
-f Suppresses printing of header line on output (the first line that defines the fields that are being displayed).
-h Suppresses printing of **.project** files on long and brief long formats.
-i Gives a quick listing with idle times.
-l Gives a long-form listing.
-m Assumes that the *User* parameter specifies a user ID (used for discretionary access control), *not* a user login name.
-p Suppresses printing of **.plan** files on long-form and brief long-form formats.
-q Gives a quick listing.
-s Gives a short format list.
-w Gives a narrow, short-format list.

Parameters

@Host	Specifies all logged-in users on the remote host.
User	Specifies a local user ID (used for discretionary access control) or local user login name, as specified in the **/etc/passwd** file.
User@Host	Specifies a user ID on the remote host, displayed in long format.

Examples

1. To get information about all users logged in to host `alcatraz`, enter:

   ```
   f @alcatraz
   ```

   Information similar to the following is displayed:

```
[alcatraz.austin.ibm.com]
Login     Name        TTY Idle         When      Site Info
brown    Bob Brown   console   2d   Mar 15 13:19
smith    Susan Smith    pts0  11:   Mar 15 13:01
jones    Joe Jones      tty0    3   Mar 15 13:01
```

User `brown` is logged in at the `console`, user `smith` is logged in from pseudo teletype line `pts0`, and user `jones` is logged in from `tty0`.

2. To get information about user `brown` at `alcatraz`, enter:

   ```
   f brown@alcatraz
   ```

   Information similar to the following is displayed:

   ```
   Login name: brown
   Directory: /home/brown    Shell: /home/bin/xinit -L -n Startup
   On since May 8 07:13:49 on console
   No Plan.
   ```

3. To get information about user `brown` at a local host in short form, enter:

   ```
   f -q brown
   ```

   Information similar to the following is displayed:

   ```
   Login           TTY             When
   brown           pts/6           Mon Dec 17 10:58
   ```

Files

/usr/bin/f	Contains the **f** command.
/etc/utmp	Contains list of users currently logged in.
/etc/passwd	Defines user accounts, names, and home directories.
/etc/security/passwd	Defines user passwords.
/var/adm/lastlog	Contains last login times.
$HOME/.plan	Optional file that contains a one-line description of a user's plan.
$HOME/.project	Optional file that contains a user's project assignment.

Related Information

The **hostname** command, **rwho** command, **finger** command.

The **fingerd** daemon.

Command for displaying information about logged-in users in *Networks and communication management*.

Communications and networks in *Networks and communication management*.

---

factor Command

Purpose

Factors a number.

Syntax

factor [*Number*]

Description

When called without specifying a value for the *Number* parameter, the **factor** command waits for you to enter a positive number less than `1E14` (100,000,000,000,000). It then writes the prime factors of that number to standard output. It displays each factor the proper number of times. To exit, enter `0` or any nonnumeric character.

When called with an argument, the **factor** command determines the prime factors of the *Number* parameter, writes the results to standard output, and exits.

Examples

To calculate the prime factors of 123, enter:

```
factor 123
```

The following is displayed:

```
123
     3
     41
```

Files

/usr/bin/factor	Contains the **factor** command.

Related Information

The **bc** command.

---

fc Command

Purpose

Processes the command history list.

Syntax

To Open an Editor to Modify and Reexecute Previously Entered Commands

fc [**-r**] [**-e** *Editor*] [*First* [*Last*]]

To Generate a Listing of Previously Entered Commands

fc -l [**-n**] [**-r**] [*First* [*Last*]]

To Generate a Listing of Previously Entered Commands with Time of Execution

fc -t [**-n**] [**-r**] [*First* [*Last*]]

To Re-execute a Previously Entered Command

fc -s [*Old*=*New*] [*First*]

Description

The **fc** command displays the contents of your command history file or invokes an editor to modify and reexecutes commands previously entered in the shell.

The command history file lists commands by number. The first number in the list is selected arbitrarily. The relationship of a number to its command does not change except when the user logs in and no other process is accessing the list. In that case, the system resets the numbering to start the oldest retained command at 1.

If the numbers in the command history file reach a limit greater than the value of the **HISTSIZE** environment variable or 32767, whichever is greater, the shell wraps to 1. Despite this optional number wrapping, the **fc** command maintains the time-ordering sequence of the commands. For example, if three commands in sequence are given the numbers 32766, 32767, and 1 (wrapped), command 32767 is still considered previous to command 1.

The commands in the history file can be displayed using the **-l** (lowercase L) flag. When the **-l** flag is not specified and commands are edited using the **-e** *Editor* flag, the resulting lines are entered at the end of the history file and then reexecuted by the shell (the **fc -e** *Editor* command is not entered into the command history list). If the editor returns a non-zero exit status, this suppresses entry in the history file and command reexecution.

Any command-line variable assignments or redirection operators used with the **fc** command again invoke the previous command, suppressing standard error for both the **fc** command and the previous command. For example:

```
fc -s -- -1 2>/dev/null
```

Flags

-e *Editor* Edits commands using the specified editor. The *Editor* parameter should be a command name. The command is located using the **PATH** environment variable. The value in the **FCEDIT** environment variable is used as a default when the **-e** flag is not specified. If the **FCEDIT** environment variable is null or unset, the ed editor is used.

-l (lowercase L) Lists the commands in your history file. No editor is invoked to modify them. The commands are written in the sequence indicated by the *First* and *Last* parameters, as affected by the **-r** flag, with each command preceded by the command number.

-n Suppresses command numbers when used with the **-l** flag.

-r Reverses the order of the commands listed (when used with the **-l** flag) or reverses the order of the commands edited (when the **-l** flag is not specified).

-s Reexecutes a command without invoking an editor. If the *First* parameter is not also specified, the **-s** flag re-executes the previous command.

-t Lists the commands in your history file along with there time of execution. The working is similar to **-l** flag but the time of execution of the command is displayed.
Note: If the time field is recorded previously by setting `EXTENDED_HISTORY=ON`, then formatted time field is displayed, else "?".

Parameters

First or *Last* Selects the commands to list or edit. The number of previous commands that can be accessed is determined by the value of the **HISTSIZE** environment variable. The *First* and *Last* parameters must have one of the following values:

[+] *Number*
Represents a specific command number. Command numbers can be displayed with the **-l** flag. A + (plus sign) is the default.

-*Number*
Represents a command that was previously executed, specified by the number of commands to back up in the history list. For example, -1 indicates the immediately previous command.

String Indicates the most recently entered command that begins with the specified string. If the *Old***=***New* parameter is specified without the **-s** flag, the string from the *First* parameter cannot contain an embedded **=** (equal sign).

When using the **-s** flag, omission of the *First* parameter causes the previous command to be used.

When the **-s** flag is not specified, the following rules apply:

- When using the **-l** flag, omission of the *Last* parameter causes a default to the previous command.
- When using the **-r**, **-n**, and **-e** flags, omission of the *Last* parameter causes a default to the *First* parameter.
- If both the *First* and *Last* parameters are omitted, the previous 16 commands are listed or the previous single command is edited (depending on whether or not the **-l** flag is used).
- If both the *First* and *Last* parameters are present, all commands are listed (when the **-l** flag is specified) or edited (when the **-l** flag is not specified). Editing multiple commands is accomplished by presenting to the editor all the commands at one time, each command starting on a new line. If the *First* parameter represents a newer command than the *Last* parameter, the commands are listed or edited in reverse sequence. This is equivalent to using the **-r** flag. For example, the following commands on the first line are equivalent to the corresponding commands on the second line:

  ```
  fc  -r  10  20          fc      30  40
  fc      20  10          fc  -r  40  30
  ```

- When a range of commands is used, it is not an error to specify *First* or *Last* values that are not in the history list. The **fc** command substitutes the value representing the oldest or newest command in the list, as appropriate. For example, if there are only ten commands in the history list, numbered 1 to 10, the commands:

  ```
  fc  -l
  fc   1   99
  ```

  list and edit, respectively, all ten commands.

Old=*New* In commands to be reexecuted, replaces the fist occurrence of the old string with the new string.

Environment Variables

The following environment variables affect the execution of the **fc** command:

EXTENDED_HISTORY Used to control the recording of time of command execution in the history file. If the variable is set to `ON` then the time is recorded, otherwise, it is not recorded.

FCEDIT When expanded by the shell, determines the default value for the **-e** *editor* variable. If the **FCEDIT** environment variable is null or is not set, the ed editor is the default.

HISTDATEFMT This is used to control the format of the time displayed by the **fc –t** command. For example, if `HISTDATEFMT=%Y`, then **fc -t** will display the year when the command is executed. The formatting is similar to that done by **date** command.

HISTFILE Determines the path name of the command history file. If the **HISTFILE** environment variable is not set, the shell may attempt to access or create the **.sh_history** file in the user's home directory.

HISTSIZE Determines a decimal number representing the limit to the number of previous commands that are accessible. If this variable is not set, a default value of128 is used.

Exit Status

The following exit values are returned:

0 Successful completion of the listing.

>0 An error occurred.

Otherwise, the exit status is that of the commands executed by the **fc** command.

Examples

1. To invoke the editor defined by the **FCEDIT** environment variable on the most recent command (the default editor is **/usr/bin/ed**), enter:

```
fc
```

The command is executed when you finish editing.

2. To list the previous two commands that were executed, enter:
   ```
   fc -l -2
   ```
3. To find the command that starts with `cc`, change `foo` to `bar`, and display and execute the command, enter:
   ```
   fc -s foo=bar cc
   ```
4. To list the previously executed commands along with there time of execution, type:
   ```
   fc –t
   ```

Files

/usr/bin/ksh	Contains the Korn shell **fc** built-in command.
/usr/bin/fc	Contains the **fc** command.

Related Information

The **ksh** command.

---

fccheck Command

Purpose

Performs basic problem determination on the First Failure Data Capture (FFDC) utilities.

Syntax

/usr/sbin/rsct/bin/fccheck [**-q**] | [**-h**]

Description

fccheck performs basic problem determination for the First Failure Data Capture utilities. The command checks for the following conditions and information on the local node:

- Checks if FFDC Error Stack usage has been disabled in the current process environment.
- Obtains the IP address that would be currently used by FFDC to identify the local node.
- Checks if **/var/adm/ffdc/stacks** is available, and if so, how much space is available in the file system where the directory resides. Checks to see if there is insufficient space to create FFDC Error Stacks.
- Checks if **/var/adm/ffdc/dumps** is available, and if so, how much space is available in the file system where the directory resides.

Results of these tests are displayed to standard output unless the ″quiet″ option has been specified. **fccheck** sets an exist status value to indicate the most severe condition it detected during the execution of its tests.

Flags

-h Displays help and usage information to standard output. No other processing is performed.

-q Specified ″quiet″ mode. The command does not display the results of each test to standard output. The exit status of the command must be used to determine the results of the tests. If more than one condition was detected, the exit status will reflect the most severe condition detected by **fccheck**.

Exit Status

The following integer exit status codes can be generated by this command:

0 All conditions tested by **fccheck** were found to be in normal operational parameters.

2 Help information successfully displayed. No further processing is performed.

12 No checking performed. Invalid option specified to this command.

19 The directory **/var/adm/ffdc/stacks** is not mounted or does not exist.

20 Cannot access or examine one or more directories in the path **/var/adm/ffdc/stacks**. Permissions may have been changed on one or more of the directories in this path to prevent access.

24 Cannot access or examine one or more directories in the path **/var/adm/ffdc/dumps**. Permissions may have been changed on one or more of the directories in this path to prevent access.

32 The directory **/var/adm/ffdc/dumps** is not mounted or does not exist.

40 Insufficient space is available in the **/var/adm/ffdc/stacks** directory to create FFDC Error Stacks on the local node.

41 Unable to obtain file system information from the operating system. This indicates a potential problem with the operating system itself.

42 FFDC Error Stack creation and usage has been disabled in this process environment.

Examples

To check for possible problems with the FFDC utilities on the local node:

```
fccheck
fccheck Status: All tests completed
```

If the local node had disabled the creation of FFDC Error Stacks, **fccheck** would indicate this as a problem:

```
fccheck

fccheck Status: Creation and use of FFDC Error Stacks has been expressly
disabled in the current execution environment. Any processes created in
the current execution environment cannot create their own FFDC Error Stacks
or inherit use of existing FFDC Error Stacks.

fccheck Status: All checks completed. Examine the previous status output for
possible FFDC problem conditions and take the recommended actions listed in
these messages.
```

Related Information

Commands: **fcclear**, **fcinit**

---

fcclear Command

Purpose

Removes FFDC Error Stacks and detail data files from the local node.

Syntax

/usr/sbin/rsct/bin/fcclear -h | [**-d** *filename* [,*filename*,...]] [**-D** *filename* [,*filename*,...]] [**-f** *FFDC_Failure_ID* [,*FFDC_Failure_ID*,...]] [**-F** *FFDC_Failure_ID* [,*FFDC_Failure_ID*,...]] [**-s** *file_name*[,*filename*,...]] [**-S** *file_name* [,*filename*,...]] [**-t** *days*]]

Description

fcclear is used to remove FFDC Error Stack files that are no longer needed for problem determination efforts from the local node. Specific FFDC Error Stack files can be removed, as well as FFDC Error Stack files containing the records of specific FFDC Failure Identifiers. Individual entries within an FFDC Error Stack cannot be removed.

Using the **-t** option, **fcclear** can be used to remove FFDC Error Stack files older than a specific number of days. To use **fcclear** in an automatic fashion to clean out unneeded FFDC Error Stacks, see the **cron** command for automating the execution of commands.

To remove all FFDC Error Stacks from the local node, specify a value of zero (0) for the number of days option argument.

Flags

-d Removes detail data files by specifying a list of one or more detail data file names. These file names may be absolute path names, or relative to the **/var/adm/ffdc/dumps** directory. These files are removed if they exist on the local node. Files on remote nodes cannot be removed through this command. If more than one file name is provided, they must be separated by a comma (,) without any intervening white space.

-D Preserves detail data files by specifying a list of one or more detail data file names. These file names may be absolute path names, or relative to the **/var/adm/ffdc/dumps** directory. These files are retained if they exist on the local node. Files on remote nodes cannot be retained through this command. If more than one file name is provided, they must be separated by a comma (,) without any intervening white space.

-f Removes FFDC Error Stack files by specifying a list of one or more FFDC Failure Identifiers. The FFDC Error Stacks associated with these FFDC Error Identifiers are located and removed if they are present on the local node. FFDC Error Stacks on remote nodes will not be removed. If more than one FFDC Failure Identifier is supplied, they must be separated by a comma (,) with no intervening white space.

-F Preserves FFDC Error Stack files by specifying a list of one or more FFDC Failure Identifiers. The FFDC Error Stacks associated with these FFDC Error Identifiers are located and retained if they are present on the local node. FFDC Error Stacks on remote nodes will not be retained. If more than one FFDC Failure Identifier is supplied, they must be separated by a comma (,) with no intervening white space.

-h Displays help and usage information to the standard output device. No other processing is performed.

-s Removes FFDC Error Stack files by specifying a list of one or more FFDC Error Stack file names. These file names can be absolute path names or file names relative to the **/var/adm/ffdc/stacks** directory. These files are removed if they exist on the local node. FFDC Error Stacks on remote nodes cannot be removed through this command. If more than one file name is provided, each must be separated by a comma (,) without any intervening white space.

-S Removes FFDC Error Stack files by specifying a list of one or more FFDC Error Stack file names. These file names can be absolute path names or file names relative to the **/var/adm/ffdc/stacks** directory. These files are removed if they exist on the local node. FFDC Error Stacks on remote nodes cannot be removed through this command. If more than one file name is provided, each must be separated by a comma (,) without any intervening white space.

-t Indicates that FFDC Error Stacks and detail data files that are older than a specific number of days should be removed from the local node. This selection criteria is independent of the other selection criteria.

Exit Status

fcclear generates the following exit status values upon completion:

0 Successful completion of the command. The command may complete successfully if no FFDC Error Stack files or detail data files match the selection criteria.

2 Help information successfully displayed. No further processing is performed.

10 No files are removed from the local system. A required option was not specified to this command.

11 No files are removed from the local system. The argument of the -t option is not numeric.

12 No files are removed from the local system. Unknown option specified by the caller.

19 The directory **/var/adm/ffdc/stacks** does not exist or is not mounted.

26 No files are removed from the local system. The same option was specified more than once.

28 No files were removed from the system. The caller provided options that instruct the command to both remove and retain the same file. This condition can occur when the command user specified an FFDC Failure Identifier that is recorded in an FFDC Error Stack file specified by name to this command.

Examples

To remove any FFDC Error Stack and detail data files older than seven days from the local node:

```
fcclear -t 7
```

To remove all FFDC Error Stack and detail data files older than seven days, but retain the FFDC Error Stack that contains information for the FFDC Failure Identifier **/3Iv04ZVVfvp.wtY0xRXQ7.....**, issue the command:

```
fcclear -t 7 -F /3Iv04ZVVfvp.wtY0xRXQ7.....
```

To remove the FFDC Error Stack file that contains the record for the FFDC Failure Identifier **/3Iv04ZVVfvp.wtY0xRXQ7.....**, issue the command:

```
fcclear -f /3Iv04ZVVfvp.wtY0xRXQ7.....
```

To remove the FFDC Error Stack files **myprog.14528.19990204134809** and **a.out.5134.19990130093256** from the system, plus the detail data file **myprog.14528.19990204135227**:

```
fcclear -s myprog.14528.19990204134809,a.out.5134.19990130093256
    -d myprog.14528.19990204135227
```

To extend the previous command to remove the named files plus any FFDC Error Stack and detail data files older that 14 days:

```
fcclear -s myprog.14528.19990204134809,a.out.5134.19990130093256
    -d myprog.14528.19990204135227 -t 14
```

Related Information

Commands: **fccheck**, **fcreport**, **fcstkrpt**

fcdecode Command

Purpose

Translates a First Failure Data Capture (FFDC) Failure Identifier from its standard form into its component parts, displaying this information to the standard output device in human readable format.

Syntax

/usr/sbin/rsct/bin/fcdecode *FFDC_Failure_ID* [,*FFDC_Failure_ID*,...] | **-h**

Description

fcdecode decodes the 42-character FFDC Failure Identifier into its component parts, and displays these parts in human readable format. The output of this command displays the following information, extracted from the FFDC Failure Identifier:

- The network address (in ASCII format) of the node where this report resides
- The time when this recording was made, expressed using the currently active time zone settings
- One of the following, depending on where the information is recorded:
- The AIX Error Log template ID used to make this recording, if the record was filed in the AIX Error Log on that node, or
- The name of the FFDC Error Stack file containing this recording, if the record was file in the FFDC Error Stack and the FFDC Error Stack resides on this node
- A suggested command that can be used to obtain the specific report associated with this FFDC Failure Identifier.

Flags

-h Displays a help message to standard output and exits. No other processing is performed, regardless of the options specified.

Parameters

FFDC_Failure_ID

An FFDC Failure Identifier, returned from previous calls to the **fcpushstk** and **fclogerr** commands, or returned from previous calls to the **fc_push_stack** or **fc_log_error** subroutines. This identifier indicates an entry made to report a failure or other noteworthy incident. More than one FFDC Failure Identifier can be provided as an argument to this command, however, each identifier must be separated by a comma (,) with no intervening white space between the identifiers.

Exit Status

fcdecode returns one of the following integer status codes upon completion:

0 FFDC Failure Identifier successfully decoded.

2 Help information displayed and processing ended.

10 An FFDC Failure Identifier was not provided as an argument to this command.

12 Invalid or unsupported option provided to this command.

27 No information written to the standard output device. The FFDC Failure Identifier argument was not valid.

Examples

The FFDC Failure Identifier is represented by a base-64 value, read from right to left. Each dot represents a leading zero. To decode the FFDC Failure Identifier **.3Iv04ZVVfvp.wtY0xRXQ7.**.....into its component parts:

```
fcdecode .3Iv04ZVVfvp.wtY0xRXQ7...................

Information for First Failure Data Capture identifier
        .3Iv04ZVVfvp.wtY0xRXQ7...................
        Generated by the local system
        Generated Thu Sep 3 11:40:17 1998 EDT
        Recorded to the AIX Error Log using template 460bb505
        To obtain the AIX Error Log information for this entry, issue
        the following command on the local system:
        TZ=EST5EDT errpt -a -j 460bb505 -s 0903114098 | more
        Search this output for an AIX Error Log entry that contains
        the following ERROR ID code:
         .3Iv04ZVVfvp.wtY0xRXQ7...................
```

The same command run on a different node has the following results:

```
fcdecode .3Iv04ZVVfvp.wtY0xRXQ7...................

Information for First Failure Data Capture identifier
        .3Iv04ZVVfvp.wtY0xRXQ7...................
        Generated on a remote system with the following Internet address:
          9.114.55.125
        Generated Thu Sep 3 11:40:17 1998 EDT
        Recorded to the AIX Error Log using template 460bb505
        TZ=EST5EDT errpt -a -j 460bb505 -s 0903114098 | more
        Search this output for an AIX Error Log entry that contains
        the following ERROR ID code:
         .3Iv04ZVVfvp.wtY0xRXQ7...................
```

Related Information

Commands: **fcdispfid**, **fcreport**, **fcstkrpt**

---

fcdispfid Command

Purpose

Displays the First Failure Data Capture Failure Identifier (FFDC Failure Identifier) to the standard error device.

Syntax

/usr/sbin/rsct/bin/fcdispfid [**-q**]*FFDC_Failure_ID* | **-h**

Description

This command is used by scripts to display an FFDC Failure Identifier value to the standard error device. This interface is provided because script programs do not have a mechanism for passing data back to its client except through exit status codes, signals, standard output, and standard error. To accomplish the task of ″passing back″ an FFDC Failure Identifier to a client in such an environment, **fcdispfid** uses XPG/4 cataloged message number **2615-000** to display this information to the standard error device. Clients of the script can capture the standard error information, search for the specific message number, and obtain the FFDC Failure Identifier from the script.

The script must indicate that any FFDC Failure Identifiers generated by the script will be directed to the standard error device in the script's user documentation. The client cannot be expected to know this behavior by default.

Flags

-h Displays a help message to standard output and exits. No other processing is performed, regardless of the options specified.

-q Suppresses warning messages from this command. If this option is not provided, this command will display messages when an invalid FFDC Failure Identifier is detected.

Parameters

FFDC_Failure_ID

Specifies an FFDC Failure Identifier. This is an identifier returned from a previous call to **fcpushstk** or **fclogerr**, and indicates an entry made to report a failure encountered by the script. This identifier is written to the standard error device using FFDC message **2615-000**.

Exit Status

0 FFDC Failure Identifier displayed to standard error.

2 Help information displayed and processing ended.

12 No information written to the standard error device. An invalid option was specified.

27 No information written to the standard error device. The *FFDC_Failure_ID* argument does not appear to be in a valid format.

Examples

To display an FFDC Failure Identifier to the client through the standard output device:

```
   FID=$(fclogerr -e FFDC_ERROR -t ERRID_SP_FFDCEXMPL_ER -i /usr/lpp/ssp/inc/
myprog.h -r myprog -s myprog.ksh -p $LINEPOS -v "1.1" -l PSSP -d $MINUSDOPTS -x
$MINUSXOPTS -y $MINUSYOPTS -b "myprog Configuration Failure - Exiting")
    RC=$?
    if ((RC == 0))
    then
         fcdispfid $FID
         return 1
    else
         :
    fi
```

Related Information

Commands: **fcdecode**, **fcfilter**, **fclogerr**, **fcpushstk**, **fcreport**, **fcstkrpt**

Subroutines: **fc_display_fid** in the *RSCT First Failure Data Capture Programming Guide and Reference*

---

fcfilter Command

Purpose

Locates and displays any First Failure Data Capture (FFDC) Failure Identifiers in a file or in standard input. More than one file may be specified.

Syntax

/usr/sbin/rsct/bin/fcfilter [*file_name*] [. . .]

Description

This commands scans any files listed as arguments for First Failure Data Capture (FFDC) Failure Identifiers. If a file name is not provided as an argument, this command examines standard input for FFDC Failure Identifiers. If an FFDC Failure Identifier is detected, **fcfilter** displays the identifier to standard output on its own line.

fcfilter can be used by scripts to extract FFDC Failure Identifiers returned by child processes via the standard error device.

If **fcfilter** detects more than one FFDC Failure Identifier in the input, the command will display all FFDC Failure Identifiers found, each one on a separate output line.

Flags

-h Displays a help message to standard output and exits. No other processing is performed, regardless of the options specified.

-q Suppresses warning messages from this command. If this option is not provided, this command will display messages when an invalid FFDC Failure Identifier is detected.

Parameters

file_name

The name of the file to be searched for an FFDC Failure Identifier. More than one file may be provided. If a file name is not provided, **fcfilter** reads from standard input.

Exit Status

fcfilter returns the following integer status codes upon completion:

0 **fcfilter** completed its execution. This exit status does not necessarily mean that any FFDC Failure Identifiers were detected.

> 0 **fcfilter** was interrupted or stopped by a signal. The exit status is the integer value of the signal that stopped the command.

Examples

The FFDC Failure Identifier is represented by a base-64 value, read from right to left. Each dot represents a leading zero. To obtain the list of all FFDC Failure Identifiers generated by a run of the command *mycmd*:

```
 mycmd 2> /tmp/errout
 fcfilter /tmp/errout
/.00...JMr4r.p9E.xRXQ7.....................
/.00...JMr4r.pMx.xRXQ7.....................
```

To obtain the FFDC Failure Identifier from a child process in a parent script, the script can use the **fcfilter** command as follows:

```
RESULTS=$(mychild 2> /tmp/errout)
if (($? != 0))               # mychild ended in failure, get FFDC ID
then
```

```
    cat /tmp/errout | fcfilter | read FIRST_FFDCID
else
    rm -f /tmp/errout
fi
```

Related Information

Commands: **fcdispfid**, **fclogerr**, **fcpushstk**, **fcreport**, **fcstkrpt**

Subroutines: **fc_display_fid**, **fc_log_error**, **fc_push_stack** (see the *RSCT First Failure Data Capture Programming Guide and Reference*)

---

fcinit Command

Purpose

Establishes or inherits a First Failure Data Capture execution environment.

Syntax

For Bourne and Korn shells:

/usr/sbin/rsct/bin/fcinit.sh [[**-l**] [**-s** {**c** | **i** }]] | [**-h**]

For C shells:

source /usr/sbin/rsct/bin/fcinit.csh [[**-l**] [-s { **c** | **i** }]] | [**-h**]

Description

This interface must be used by a script program that wishes to use the FFDC interfaces for recording information to the AIX Error Log, the BSD System Log, or the FFDC Error Stack .

Applications may wish to establish an FFDC Environment for one of the following reasons:

- The script may wish to record information to the AIX Error Log. Scripts can use **fcinit** to establish a basic FFDC Environment
- The script wants to have itself and any descendant processes created by itself or its children to record failure information to the FFDC Error Stack. In this case, the script considers itself a ″top-level″ application that will cause multiple ″lower-level″ applications to be created, and the success of the ″top-level″ application depends upon the success of these ″lower-level″ applications. When using **fcinit** in this fashion, the process is said to *establish* or *create* the FFDC Error Stack Environment.
- The script uses the FFDC Error Stack or the FFDC Trace only in those cases when the script is invoked by an ancestor process that wants failure information or trace information recorded to these devices. In all other cases, the script does not wish to use these devices. When using **fcinit** in this fashion, the process is said to *inherit* the FFDC Error Stack Environment.

Any process wishing to record information to the AIX Error Log or the BSD System Log through the FFDC interfaces must establish an FFDC Environment. If the process does not wish to make use of an FFDC Error Stack, the process can establish a basic FFDC Environment that does not make use of an FFDC Error Stack. An FFDC Error Stack Environment, which contains an FFDC Error Stack, is established by a process when that process wants to have failure information from itself, any threads it may create, and any descendant processes it may create to be recorded in an FFDC Error Stack. An *FFDC Error Stack Environment*, which contains an FFDC Error Stack, is inherited by a process when that process wants to record failure information to an FFDC Error Stack file only when one of its ancestors has requested for processes to do so; in all other cases, the process will not record failure information to the FFDC Error Stack.

The FFDC Error Stack Environment, which contains an FFDC Error Stack, reserves an FFDC Error Stack file, so that failure information is recorded to a file in the **/var/adm/ffdc/stacks** directory. These files use the naming format ***script_name.PID.date_and_time***, where *script_name* is the name of the script itself, *PID* is the process identifier of the script, and *date_and_time* is the date and time when the script was executed. Whenever this script or children processes of this script record failure information to the FFDC Error Stack, it will be recorded in this file.

In order for information to be recorded in the FFDC Error Stack by a process, the process must use the **fcpushstk** FFDC interface, and the process has to be operating within an established FFDC Error Stack Environment. If an FFDC Error Stack Environment does not exist, or if the **fcpushstk** interface is not used when an FFDC Error Stack Environment exists, no information is recorded by that process in the FFDC Error Stack. This function permits processes to run in a normal or ″silent″ mode when failure debugging information is not wanted or needed, but also permits this information to be available when the process is invoked within a special environment for debugging.

fcinit must be executed within the FFDC client's process environment (″sourced″) in order for the command to properly set the FFDC Environment for the script. Script-based FFDC clients using this command must ″source″ the command in order for **fcinit** to execute within the client's process image. If this is not done. the FFDC interface is executed within its own process image; any settings of the FFDC Environment are lost after the FFDC interface completes. To demonstrate how a script-based application would ″source″ the **fcinit** command, a Korn Shell program would issue the following instruction:

```
. fcinit.sh <options and arguments>
```

A C Shell script would do the following:

```
source fcinit.csh <options and arguments>
```

Processes that use the **fclogerr** FFDC interface must establish an *FFDC Environment*. If the process only wishes to use the **fclogerr** interface, the *FFDC Environment* can be established without an FFDC Error Stack.

If an FFDC Environment already exists when a script attempts to create one, the script inherits the existing FFDC Environment instead of creating its own.

Flags

-h Displays a help message to standard output and exits. No other processing is performed, regardless of the options specified.

-l Indicates that the process wishes to make use of the AIX Error Log only. This option is not necessary when the **-s** option is specified, since use of the AIX Error Log is permitted within an FFDC Error Stack Environment.

-s Indicates that an FFDC Error Stack Environment is to be established. Applications wishing to use the **fcpushstk** interface must specify this flag. Upon successful completion of this command, an FFDC Error Stack file is reserved for the script in the **/var/adm/ffdc/stacks** directory. This flag must be specified with one of two possible options:

- **c** Requests that the FFDC Error Stack Environment be *created*. If an FFDC Error Stack Environment was not created by an ancestor process, it will be created. If such an environment was previously created by an ancestor process, this process will *inherit* the FFDC Error Stack Environment as if the **i** option had been specified.
- **i** Specifies that an FFDC Error Stack Environment is to be *inherited* if it was previously established by an ancestor process. If an FFDC Error Stack Environment was not previously established by an ancestor process, an FFDC Error Stack Environment is not

established for this process, and this process cannot make use of an FFDC Error Stack (although it may make use of the AIX Error Log and the BSD System Log).

Parameters

file_name
The name of the file to be searched for an FFDC Failure Identifier. More than one file may be provided. If a file name is not provided, **fcfilter** reads from standard input.

Exit Status

fcinit returns the following exit status codes upon completion:

0 FFDC Environment successfully established.

1 FFDC Environment successfully inherited.

2 Help information displayed and processing ended.

fcinit returns the following exit status codes upon detection of a failure:

12 FFDC Environment not established or inherited - Unknown function parameter provided.

13 FFDC Error Stack Environment not established or inherited - caller indicated that the FFDC Environment should be both created and inherited.

14 FFDC Environment not established in this call - the caller already has an FFDC Environment established for itself - this routine may have been executed multiple times.

15 FFDC Error Stack Environment not established or inherited - an FFDC Error Stack Environment did not exist, and the FC_INHERIT option was specified.

16 FFDC Environment not established or inherited - the client's process environment could not be modified by this routine.

17 FFDC Environment not established or inherited - the FFDC Environment appears to be corrupted and should be considered unusable.

18 FFDC Environment not established or inherited - the routine could not allocate the memory required to modify the client's process environment.

19 FFDC Error Stack Environment not established or inherited - Unable to reserve the FFDC Error Stack file for the calling process - the FFDC Error Stack directory does not exist or cannot be used.

21 FFDC Error Stack Environment not established or inherited - Unable to reserve the FFDC Error Stack file for the calling process - the file already exists

42 FFDC Error Stack Environment not established or inherited - creation and use of FFDC Error Stacks has been disabled by the system administrator. Scripts can establish only a basic FFDC Environment that makes use of the AIX Error Log and the BSD System Log.

99 FFDC Environment not established or inherited - an unexpected internal failure occurred within **fcinit**. This condition may require the attention of customer and application-support services.

Examples

For a Korn Shell script to establish a basic FFDC Environment for using the AIX Error Log and the BSD System Log only (an FFDC Error Stack is not to be used or reserved):

```
# Set up an FFDC Environment to use the AIX Error Log only. An FFDC Error
# Stack is not needed for this script.
. fcinit.sh -1
rc=$?
```

```
if ((rc != 0))
    then
         print "fcinit failed with exit code of $rc"
         exit 1
fi
# Normal processing starts
```

For a Korn Shell script to establish an FFDC Error Stack Environment that causes the script and any descendant process to record failure information to the FFDC Error Stack:

```
# Set up FFDC Environment to record failure information to the FFDC Error
# Stack
. fcinit.sh -sc
rc=$?
if ((rc != 0))
then
    print "fcinit failed with a code of $rc"
    exit 1
fi
# Normal processing starts
```

Note: The FFDC client may receive an indication that an FFDC Error Stack Environment was inherited, instead of created by the **fcinit** call. This occurs when an FFDC Error Stack Environment was already established by one of the process's ancestors.

To inherit an FFDC Error Stack Environment from the process's parent process:

```
# Inherit an FFDC Environment from parent process if it exists - otherwise,
# operate in a normal "silent" mode
. fcinit.sh -si
rc=$?
if ((rc != 0))
then
    print "fcinit failed with a code of $rc"
    exit 1
fi
# Normal processing starts
```

Related Information

Commands: **fccheck**, **fclogerr**, **fcpushstk**, **fcteststk**

Subroutines: **fc_init** (see the *RSCT First Failure Data Capture Programming Guide and Reference*)

---

fclogerr Command

Purpose

Records information about failure or noteworthy conditions to the AIX error log and the BSD system log.

Syntax

/usr/sbin/rsct/bin/fclogerr { **-e** *event* **-t** *error_template_label* **-i** *error_template_headerfile* **-r** *resource* **-s** *source_filename* **-p** *line_of_code_pos* **-v** *sidlevel* **-l** *lpp_name* **-a** *assoc_fid* { [**-d** *detail_data_item*[,*detail_data_item*,...] **-x** *detail_data_type*[,*detail_data_type*,...] **-y** *detail_data_len*[,*detail_data_len*,...]] | [**-f** *detail_data_file*] } **-b** *BSD_syslog_message_text* } | **-h**

Description

This interface is used by any script program that wishes to record information to the AIX Error Log and the BSD System Log. The information written to this device is intended for use by the system administrator or operator to determine what failure conditions or other noteworthy conditions have occurred on the system that require attention. The purpose of the AIX Error Log and the BSD System Log is to record enough information about a condition so that the nature, impact, and response to the condition can be determined from the report, without requiring a recreation of the condition to detect what condition occurred and where. Any software that encounters permanent failure conditions that will persist until some type of direct intervention occurs, or encounters a condition that should be brought to the attention of the system administrator, should use **fclogerr** to record this information in the AIX Error Log and the BSD System Log.

Scripts should establish a basic FFDC Environment or an FFDC Error Stack Environment before using **fclogerr**, either by creating or inheriting the environment. **fclogerr** records information to the AIX Error Log and the BSD System Log even if these environments are not established, but the interface will not be capable of generating an FFDC Failure Identifier unless one of these environments exists.

Processes designed to use the FFDC Error Stack can also make use of the **fclogerr** interface, and should make use of it if they encounter conditions that require administrator attention or intervention to resolve.

To ensure proper identification of the condition and the location at which it was encountered, the FFDC Policy recommends that **fclogerr** should be called in-line in the script's source code module and invoked as soon as the condition is detected. **fclogerr** will record source code file name and line of code information to assist in identifying and locating the source code that encountered the condition. **fclogerr** can be invoked by a subroutine or autoloaded routine to record this information if this is necessary, provided that all location information and necessary failure detail information is made available to this external routine. The external recording routine must record the true location where incident was detected.

Although **fclogerr** reports information to both the AIX Error Log and the BSD System Log, different options must be provided to this interface for each recording device. The Detail Data information recorded to the AIX Error Log is not also recorded to the BSD System Log; BSD System Log information is provided through different command options. This may require the **fclogerr** user to duplicate some information in this call.

Flags

-a Contains the FFDC Failure Identifier for a failure condition reported by software used by this application which causes or influenced the condition being recorded at this time. This identifier should have been returned to this application as part of the software's result indication. The caller provides this identifier here so that the FFDC Error Stack can associate the failure report it is making at this time with the previously recorded failure report. This permits problem investigators to trace the cause of a failure from its various symptoms in this application and others to the root cause in the other software. If no other software failure is responsible for this condition, or if the other software did not return an FFDC Failure Identifier as part of its result information, this option should be omitted.

-b Specifies the text message to be written to the BSD System Log.

-d One or more data items that provides detailed information on the condition, used to provide the Detail Data in the AIX Error Log entry. If details of the information are too lengthy, these details can be written to a file, and the name of that file provided as the *detail_data_file* parameter. If a detail data file name is provided, this option should be omitted. If neither the *detail_data* or the *detail_data_file* parameters are provided or appear valid, null information will be recorded for the detail data in the AIX Error Log.

More than one data item may be provided with this option. Each data item must be separated by commas (,) with no intervening white-space characters. If a data item has imbedded whitespace

characters, the data item must be enclosed in double quotes (″). The data items themselves must not contain commas (,), as the command interprets commands a field separators.

This option *must* be accompanied by the **-x** and **-y** options.

-e Specifies the FFDC Log Event Type. Current valid values are FFDC_EMERG, FFDC_ERROR, FFDC_STATE, FFDC_TRACE, FFDC_RECOV, and FFDC_DEBUG. This code gives a general description of the type of event being logged (emergency condition, permanent condition, informational notification, debugging information, etc.) and the severity of the condition. If this option is not specified, the event type FFDC_DEBUG is assigned to this incident record.

-f Name of a file containing details about the condition being reported. This option is used when the details are too lengthy to record within the remaining 100 bytes of Detail Data information left to the application by **fclogerr**, or when a utility exists that can analyze the detail information. The contents of this file is copied to the **/var/adm/ffdc/dumps** directory, and the file's new location is recorded as the Detail Data in the AIX Error Log entry.

-h Displays a help message to standard output and exits. No other processing is performed, regardless of the options specified.

-i Specifies the absolute path name of the header file (.h) that contains the error logging template identification number that corresponds to the *error_template_label* specified in the **-l** option. This template must also be found in the node's error logging template repository (**/var/adm/ras/errtmplt**). This header file was generated by the **errupdate** command as part of the source code's building procedures, and should have been included in the LPP's packaging to be installed on the node with the software. If this option is not specified or the header file cannot be found when the script is executed, **fclogerr** will record the failure information using its own default error template (label FFDC_DEF_TPLT_TR, identifier code 2B4F5CAB).

-l Specifies an abbreviation of the name of the licensed programming product in which this software was shipped. This value should be recognizable to both customer and application-support services as an acceptable name for the LPP. Examples of such values are: PSSP, GPFS, LoadLeveler®, and RSCT. If this option is not provided or appears invalid, the character string **PPS_PRODUCT** is used.

-p Specifies the line of code location within the source code module where the condition is being reported. The value provided must be a valid integer value. To allow for proper identification and location of the condition, this value should be as close to the line of code that detected the condition as possible. Korn Shell scripts can use the value of **$LINENO**. Script languages that do not provide a special line count variable can provide a symbolic value here that a developer can use to locate the spot in the source code where **fclogerr** is being used. If this option is not valid or not provided, the value of **0** is used.

-q Suppresses the generation of warning messages from the command. Warning are generated when the command must substitute default information for missing information, or when the command is unable to copy the *detail_data_file* to the **/var/adm/ffdc/dumps** directory.

-r Specifies the software component name. This is a symbolic name for the software making the report and should be a name recognizable to both customer and application-support services. The character string is limited to 16 characters.

-s Specifies the name of the source file containing the line of code that encountered the condition being reported. For Korn and Borne Shell scripts, the argument to this option should be set to **$0**; C Shell scripts would set this argument to **${0}**. If this option is not provided or not valid, the character string **unknown_file** is used.

-t Indicates the symbolic label given to the AIX Error Logging template in the error log repository. The **errupdate** command that builds error logging templates creates a macro that maps this label to an integer code. This label begins with the characters **ERRID_** and is a maximum of 19

characters. If this option is not specified or the header file cannot be found when the script is executed, **fclogerr** will invoke the **errlogger** to create a message in the AIX Error Log using the OPMSG template.

-v Indicates the SCCS version number of the source code module that detected the condition being recorded. For source code built under SCCS control, this should be set to ″**1.1**″ (the double-quotes are necessary). If this option is not provided or is not valid, the character string **unknown** is used.

-x Indicates how the data items specified by the **-d** option are to be interpreted when recording this information to the AIX Error Log. These types must agree with the corresponding fields of the AIX Error Logging template specified in the **-t** option. Each type indicates how the corresponding data item in the **-d** list is interpreted. Acceptable values for this option are ALPHA, HEX, and DEC. There must be a matching type listed in the **-x** argument for each argument in the **-d** list.

This option *must* be supplied if the **-d** option is provided.

-y Indicates the length of the data items (in bytes) specified by the **-d** option. These lengths must agree with the corresponding fields of the AIX Error Logging template specified in the **-t** option. There must be a matching type listed in the **-y** argument for each argument in the **-d** list.

This option *must* be supplied if the **-d** option is provided.

Parameters

file_name
The name of the file to be searched for an FFDC Failure Identifier. More than one file may be provided. If a file name is not provided, **fcfilter** reads from standard input.

Exit Status

fclogerr returns the following exit status codes upon successful completion:

0 Information successfully queued to be written to the AIX Error Log and the BSD System Log. An FFDC Failure Identifier for the record is displayed to standard output. The caller should capture standard output to obtain this value.

2 Help information displayed and processing ended.

12 No information recorded to the AIX Error Log, and no FFDC Failure Identifier is provided by the command. The command user provided an invalid option to this command.

On UNIX platforms other than AIX, **fclogerr** returns the following exit status codes when a failure occurs:

38 A record could not be made int he BSD System Log for this incident. The System Log is experiencing a failure condition. On AIX systems, a report was recorded to the AIX Error Log; on other systems, this should be considered a failure.

When **fclogerr** is provided with incomplete information, it substitutes default information for the missing information and attempts to make a record in the FFDC Error Stack. Warnings are generated in these cases, and warning messages are generated unless the **-q** option is specified. In cases where more than one warning condition was detected, the command returns an exit status code for the condition it considered the most severe. The following exit status codes are returned by **fclogerr** when warning conditions are detected:

10 The command user failed to provide the **-i** option to this command, or the header file named as the argument to the -i option could not be located. The command will record generic information to the AIX Error Log in this case, using the First Failure Data Capture default template (label FFDC_DEF_TPLT_TR, identifier code 2B4F5CAB).

26 Both a detailed data string and a detail data file were provided to this routine. The routine chose the detail data string and ignored the detail data file.

28 The name of the resource detecting the incident was not provided. The default resource name **ffdc** was substituted for the missing resource name.

29 At least one component of the detecting application information—source code file name, source code file version, LPP name, line of code position—was not provided. Default information was substituted for the missing information.

32 The file named in the *detail_data_file* parameter could not be copied to the **/var/adm/ffdc/dumps** directory. The FFDC Error Stack entry cites the original version of this file. Do not discard the original copy of this file.

33 The **-e** option was not specified, or did not specify a valid FFDC event type. The event type FFDC_DEBUG has been assigned to this incident record.

34 A message was not supplied in the *format* parameter. As a result, a generic message was recorded to the BSD System Log for this incident.

35 No detailed information was provided for this incident. Later problem analysis may be difficult without these details to indicate specifics on the incident.

36 The length of the detail data string was greater than the capacity of the AIX Error Log entry limit. Detail data was truncated to fit in the available space. Some information on the incident may have been lost in this truncation.

37 An FFDC Error Identifier could not be constructed for the report created by this routine. An FFDC Failure Identifier is not written to standard output, but information on the incident was recorded to the AIX Error Log and the BSD System Log.

38 A record could not be made in the BSD System Log for this incident. The System Log may not be enabled, or may be experiencing problems. On AIX systems, a report was recorded to the AIX Error Log; on other systems, this should be considered a failure.

Examples

For this example, a Korn Shell script attempts to access configuration information from a file. If this attempt fails, the code will record a failure to the AIX Error Log using the following template source code:

```
*! mymesgcat.cat
+ SP_FFDCEXMPL_ER:
    Comment        = "Configuration Failed - Exiting"
    Class          = S
    Log       = true
    Report         = true
    Alert          = false
    Err_Type  = PERM
    Err_Desc  = {3, 10, "CONFIGURATION FAILURE - EXITING"}
    Prob_Causes    = E89B
    User_Causes    = E811
    User_Actions   = 1056
    Fail_Causes    = E906, E915, F072, 108E
    Fail_Actions   = {5, 14, "VERIFY USER HAS CORRECT PERMISSIONS TO ACCESS FILE"},
               {5, 15, "VERIFY CONFIGURATION FILE"}
    Detail_Data    = 46, 00A2, ALPHA
    Detail_Data    = 42, EB2B, ALPHA
    Detail_Data    = 42, 0030, ALPHA
    Detail_Data    = 16, EB00, ALPHA
    Detail_Data    = 16, 0027, ALPHA
    Detail_Data    = 4, 8183, DEC
    Detail_Data    = 4, 8015, DEC
    Detail_Data    = 60, 8172, ALPHA
```

This definition yields the following AIX Error Logging Template:

```
LABEL:            ERRID_SP_FFDCEXMPL_ER
IDENTIFIER:        <calculated by errupdate during source code build>

Date/Time:         <filled in by AIX Error Log subsystem>
Sequence Number:   <filled in by AIX Error Log subsystem>
Machine Id:        <filled in by AIX Error Log subsystem>
Node Id:           <filled in by AIX Error Log subsystem>
Class:             S
Type:              PERM
Resource Name:     <filled in by -r option to fclogerr>

Description
CONFIGURATION FAILURE - EXITING

Probable Causes
COULD NOT ACCESS CONFIGURATION FILE

User Causes
USER CORRUPTED THE CONFIGRATION DATABASE OR METHOD

    Recommended Actions
    RE-CREATE FILE

Failure Causes
COULD NOT ACCESS CONFIGURATION FILE
PERMISSIONS ERROR ACCESSING CONFIGURATION DATABASE
FILE READ ERROR
FILE IS CORRUPT

    Recommended Actions
    VERIFY USER HAS CORRECT PERMISSIONS TO ACCESS FILE
    VERIFY CONFIGURATION FILE

Detail Data
DETECTING MODULE
<filled in by fclogerr options>
ERROR ID
<The FFDC Failure Identifier created by fclogerr>
REFERENCE CODE
<The -a option value to fclogerr>
FILE NAME
<Must be supplied as part of -d option list to fclogerr>
FUNCTION
<Must be supplied as part of -d option list to fclogerr>
RETURN CODE<Must be supplied as part of -d option list to fclogerr>
ERROR CODE AS DEFINED IN sys/errno.h
<Must be supplied as part of -d option list to fclogerr>
USER ID<Must be supplied as part of -d option list to fclogerr>
```

The first three Detail Data Fields are constructed by the **fclogerr** routine from information passed in the parameters. The remaining Detail Data must be supplied with the **-d** option, and the type of data supplied must be indicated by the **-x** option. The example source code segment below demonstrates how this is done, and how **fclogerr** is invoked to record the information in the AIX Error Log and the BSD System Log.

```
typeset CONFIG_FNAME
typeset INBUF
typeset MINUSDOPTS
typeset MINUSXOPTS
typeset MINUSYOPTS
typeset FID
integer MYCLIENT
integer RC
    :
```

```
MYCLIENT=$$
CONFIG_FNAME="/configfile.bin"
exec 3< $CONFIG_FNAME
    :
read -u3 INBUF
RC=$?
if ((RC != 0))
then
    # Create Detail Data Memory Block for AIX Error Log Template
    # Need to know the EXACT structure of the Template to do this correctly.
    #    Field 1 - filled in by fc_log_error
    #    Field 2 - filled in by fc_log_error
    #    Field 3 - filled in by fc_log_error
    #    Field 4 - name of configuration file being used - 16 bytes
    #    Field 5 - name of function call that failed - 16 bytes
    #    Field 6 - return code from failing function - 4 byte integer
    #    Field 7 - errno from failing function call (unused) - 4 byte integer
    #    Field 8 - user ID using this software - remaining space (62 bytes)
    # This source code supplied fields 4 through 8 in the "-d" option, and
    # describes the data types for each in the "-x" option.
    MINUSDOPTS=$CONFIG_FNAME
    MINUSXOPTS="ALPHA"
    MINUSYOPTS="16"
    MINUSDOPTS="$MINUSDOPTS,read"
    MINUSXOPTS="$MINUSXOPTS,ALPHA"
    MINUSYOPTS="$MINUSYOPTS,16"
    MINUSDOPTS="$MINUSDOPTS,$RC"
    MINUSXOPTS="$MINUSXOPTS,DEC"
    MINUSYOPTS="$MINUSYOPTS,4"
    MINUSDOPTS="$MINUSDOPTS,0"
    MINUSXOPTS="$MINUSXOPTS,DEC"
    MINUSYOPTS="$MINUSYOPTS,4"
    MINUSDOPTS="$MINUSDOPTS,$MYCLIENT"
    MINUSXOPTS="$MINUSXOPTS,DEC"
    MINUSYOPTS="$MINUSYOPTS,60"
    FID=$(fclogerr -e FFDC_ERROR -t ERRID_SP_FFDCEXMPL_ER -i /usr/lpp/ssp/inc/
myprog.h -r myprog -s myprog.ksh -p $LINEPOS -v "1.1" -l PSSP -d $MINUSDOPTS -x
$MINUSXOPTS -y $MINUSYOPTS -b "myprog Configuration Failure - Exiting")
    RC=$?
    if ((RC == 0))
    then
         fcdispfid $FID
         return 1
    else
         :
    fi
fi
```

Now consider a slight variation on the above example, using the same AIX Error Logging template, but this time using an external command to obtain the configuration data from a file that this source code supplies. The command exits with a non-zero exit status and prints an FFDC Failure Identifier to standard output if it encounters any failure conditions. Also, to demonstrate the use of double-quotes in the **-d** list, the configuration file will have an embedded space in the name:

```
typeset CONFIG_FNAME
typeset INBUF
typeset MINUSDOPTS
typeset MINUSXOPTS
typeset MINUSYOPTS
typeset FID
typeset OUTPUT
integer MYCLIENT
integer RC
    :
MYCLIENT=$$
```

```
CONFIG_FNAME="This is a test"
OUTPUT=$(configdabeast $CONFIG_FNAME)
RC=$?
if ((RC != 0))
then
    # Create Detail Data Memory Block for AIX Error Log Template
    # Need to know the EXACT structure of the Template to do this correctly.
    #    Field 1 - filled in by fc_log_error
    #    Field 2 - filled in by fc_log_error
    #    Field 3 - filled in by fc_log_error
    #    Field 4 - name of configuration file being used - 16 bytes
    #    Field 5 - name of function call that failed - 16 bytes
    #    Field 6 - return code from failing function - 4 byte integer
    #    Field 7 - errno from failing function call (unused) - 4 byte integer
    #    Field 8 - user ID using this software - remaining space (62 bytes)
    # This source code supplied fields 4 through 8 in the "-d" option, and
    # describes the data types for each in the "-x" option.
    MINUSDOPTS="\""$CONFIG_FNAME"\""
    MINUSXOPTS="ALPHA"
    MINUSYOPTS="16"
    MINUSDOPTS="$MINUSDOPTS,configdabeast"
    MINUSXOPTS="$MINUSXOPTS,ALPHA"
    MINUSYOPTS="$MINUSYOPTS,16"
    MINUSDOPTS="$MINUSDOPTS,$RC"
    MINUSXOPTS="$MINUSXOPTS,DEC"
    MINUSYOPTS="$MINUSYOPTS,4"
    MINUSDOPTS="$MINUSDOPTS,0"
    MINUSXOPTS="$MINUSXOPTS,DEC"
    MINUSYOPTS="$MINUSYOPTS,4"
    MINUSDOPTS="$MINUSDOPTS,$MYCLIENT"
    MINUSXOPTS="$MINUSXOPTS,DEC"
    MINUSYOPTS="$MINUSYOPTS,60"
    FID=$(fclogerr -e FFDC_ERROR -t ERRID_SP_FFDCEXMPL_ER -i /usr/lpp/ssp/inc/
myprog.h -r myprog -s myprog.ksh -p $LINEPOS -v "1.1" -l PSSP -d $MINUSDOPTS -x
$MINUSXOPTS -y $MINUSYOPTS -a $OUTPUT -b "myprog Configuration Failure - Exiting")
    RC=$?
    if ((RC == 0))
    then
         fcdispfid $FID
         return 1
    else
         :
    fi
fi
```

Related Information

Commands: **errpt**, **fcdecode**, **fcdispfid**, **fcinit**, **fcpushstk**, **fcreport**

Files: **ct_ffdc.h**

Subroutines: **fc_log_error** (see the *RSCT First Failure Data Capture Programming Guide and Reference*)

---

fcpushstk Command

Purpose

Records information about failure or noteworthy conditions to the First Failure Data Capture Error Stack.

Syntax

/usr/sbin/rsct/bin/fcpushstk { [**-a** *assoc_fid*] **-c** *message_catalog_name* **-m** *message_set* **-n** *message_number* [**-o** *message_param* [,*message_param*,...]] **-l** *lpp_name* **-p** *line_of_code_pos* **-r** *resource* **-s** *source_filename* **-v** *sidlevel* {[**-d** *detail_data*] | [**-f** *detail_data_file*]} } *default_message* | **-h**

Description

fcpushstk is used by scripts to record failure information to the FFDC Error Stack. Scripts record descriptive information and debugging data to the FFDC Error Stack for use in later problem determination efforts.

The FFDC Error Stack is used to help understand failure conditions that occur when multiple related processes or threads are executing together on a node to perform a common task. This device is best applied to an application that creates one or more threads or subprocesses, which in turn, may also create threads or subprocesses themselves. To use the FFDC Error Stack, the script establishes an *FFDC Error Stack Environment* using the **fcinit** interface. After this environment is established, the application and any of its descendants can make use of the FFDC Error Stack.

Not all software applications will establish an FFDC Error Stack Environment. However, these applications may be invoked by other applications or scripts that establish FFDC Error Stack Environments. In these cases, the scripts or applications invoking this software may wish to capture the failure information from this software, to analyze it along with other failure information from other software it invokes to discover any relationships or patterns in the failures. For this reason, software that ordinarily would not make use of the FFDC Error Stack under normal operational conditions should at least support the use of the FFDC Error Stack when it is used by any client invoking the software. This is accomplished by *inheriting* the FFDC Error Stack Environment from the parent process through the **fcinit** interface.

fcpushstk records descriptions and details about noteworthy conditions to the FFDC Error Stack. If an *FFDC Error Stack Environment* has not been established by the script, either by creation or inheritance, **fcpushstk** does not record any information and returns control back to the caller. This action permits the script to run in a normal ″silent″ mode when debugging information is not requested, but also permits the script to support the use of the FFDC Error Stack when debugging information is requested.

Scripts must make explicit calls to **fcpushstk** to record information to the FFDC Error Stack when an FFDC Error Stack Environment is established. Merely establishing the environment is not enough to result in failure data being recorded. The **fclogerr** command will not make any records to the FFDC Error Stack.

To ensure proper identification of the condition and the location at which it was encountered, **fcpushstk** should be called in-line in the script's source code module, invoked as soon as the condition is detected. **fcpushstk** will record source code file name and line of code information to assist in identifying and locating the source code that encountered the condition. **fcpushstk** can be invoked by a subroutine or autoloaded routine to record this information if this is necessary, provided that all location information and necessary failure detail information is made available to this external routine. The external recording routine must record the true location where the incident was detected.

The maximum size of an FFDC Error Stack entry is given by the FC_STACK_MAX definition in the **<rsct/ct_ffdc.h>** header file. FC_STACK_MAX defines a length in bytes. This value should be used only as a rough guide, since this length includes data that will be used by **fcpushstk** to record the detecting file information, description information, and FFDC Failure Identifier information. Any records longer than FC_STACK_MAX bytes will be truncated to fit within the FC_STACK_MAX limit.

Flags

-a Specifies an FFDC Failure Identifier for a failure condition reported by software used by this application which causes or influenced the condition being recorded at this time. This identifier should have been returned to this application as part of the software's result indication. The caller provides this identifier here so that the FFDC Error Stack can associate the failure report it is making at this time with the previously recorded failure report. This permits problem investigators to trace the cause of a failure from its various symptoms in this application and others to the root cause in the other software. If no other software failure is responsible for this condition, or if the other software did not return an FFDC Failure Identifier as part of its result information, the **-a** option should not be provided.

-c Indicates the name of the XPG/4-compliant message catalog that contains a description of the failure being recorded. This name is relative to the **/usr/lib/nls/msg/$LANG** directory. If the message catalog cannot be found, the *default_message* will be displayed to describe the failure. Note that the *default_message* will not be translated between locales.

-d A character string that provides detailed information on the condition, similar to the Detail Data concept used by the AIX Error Log. If details of the information are too lengthy, these details can be written to a file, and the name of that file provided as an argument to the **-f** option. The **-d** and **-f** options cannot be specified at the same time. If neither the **-d** or the **-f** options are provided or appear valid, the character string **no detail data** is recorded.

-f Specifies the name of a file containing details about the condition being reported, similar to the Detail Data concept used by the AIX Error Log. This option is used when the details are too lengthy to record within the FFDC Error Stack itself, or when a utility exists that can analyze the detail information. The contents of this file is copied to the **/var/adm/ffdc/dumps** directory, and the file's new location is recorded as the Detail Data in the FFDC Error Stack. If a file containing details of the condition does not exist, do not specify this option. The **-d** and **-f** options cannot be specified at the same time.

-h Displays a help message to standard output and exits. No other processing is performed, regardless of the options specified.

-l Specifies an abbreviation of the name of the licensed programming product in which this software was shipped. This value should be recognizable to both customer and application-support services as an acceptable name for the LPP. Examples of such values are: PSSP, GPFS, LoadLeveler, and RSCT. If this option is not provided or appears invalid, the character string **PPS_PRODUCT** is used.

-m Specifies the message set containing the message describing the failure in the message catalog file. If this message set cannot be located, the *default_message* will be displayed to describe the failure. Note that **default_message** will not be translated to the user's locale.

-n Specifies the message number that describes the failure being recorded. If this message cannot be located, the *default_message* will be displayed to describe the failure. Note that *default_message* will not be translated to the user's locale.

-o Specifies a list of substitution parameters within the message indicated by the **-n** option. **fcpushstk** only supports character strings as substitutional parameters (%s) due to the shell operating environment. If multiple substitutional parameters are provided, each one must be separated by a comma (,). If any of these substitution parameters contain imbedded white space, they must be enclosed in double quotes (″″).

-q Suppresses the generation of warning messages from the command. Warning are generated when the command must substitute default information for missing information, or when the command is unable to copy the *detail_data_file* to the **/var/adm/ffdc/dumps** directory.

-r Specifies the software component name. This is a symbolic name for the software making the report, and should be a name recognizable to both customer and application-support services.

-p Specifies the line of code location within the source code module where the condition is being reported. The value provided must be a valid integer value. To allow for proper identification and location of the condition, this value should be as close to the line of code that detected the condition as possible. Korn Shell scripts can use the value of **$LINENO**. Script languages that do not provide a special line count variable can provide a symbolic value here that a developer can use to locate the spot in the source code where **fcpushstk** is being used. If this option is not valid or not provided, the value of **0** is used.

-s Specifies the name of the source file containing the line of code that encountered the condition being reported. For Korn and Borne Shell scripts, the argument to this option should be set to **$0**; C Shell scripts would set this argument to ${0}. If this option is not provided or not valid, the character string **unknown_file** is used.

-v Indicates the SCCS version number of the source code module that detected the condition being recorded. For source code under SCCS control, this should be set to ″**1.1**″ (the double-quotes are necessary). If this option is not provided or is not valid, the character string **unknown** is used.

Parameters

default_message
Indicates a default message to be used as a description of the failure, when the information cannot be retrieved from the message catalog information supplied through the **-c**, **-m**, and **-n** options. If this string contains positional parameters, all positional parameters must be specified to be character strings (%s). The message should be enclosed in double quotes (″″) if it contains any embedded white space. **fcpushstk** limits the overall length of this string to 72 characters.

Exit Status

fcpushstk returns the following exit status codes upon successful completion:

0 FFDC Error Stack Environment exists, and failure information successfully recorded in the FFDC Error Stack. An FFDC Failure Identifier for the record is displayed to standard output. The caller should capture standard output to obtain this value.

2 Help information displayed and processing ended.

fcpushstk returns the following exit status codes when a failure occurs:

11 No information recorded to the FFDC Error Stack, and no FFDC Failure Identifier is provided by this command. The client requested to use an option not supported in this release of the FFDC software

12 No information recorded to the FFDC Error Stack, and no FFDC Failure Identifier is provided by this command. Unknown function parameter provided to the interface.

15 FFDC Error Stack Environment does not exist. No information recorded to the FFDC Error Stack. No FFDC Failure Identifier is generated by this command. This is the normal return code to the FFDC client when an FFDC Error Stack Environment did not exist to be inherited via **fcinit**.

17 No information recorded to the FFDC Error Stack, and no FFDC Failure Identifier is provided by this command. The FFDC Error Stack Environment appears to be corrupted and should be considered unusable.

19 No information recorded to the FFDC Error Stack - the FFDC Error Stack directory does not exist or cannot be used. No FFDC Failure Identifier is provided by this command.

20 No information recorded to the FFDC Error Stack, and no FFDC Failure Identifier is provided by this command. Unable to access the FFDC Error Stack file. The file may have been removed, or permissions on the file or its directory have been changed to prohibit access to the FFDC Error Stack.

22 No information recorded to the FFDC Error Stack - the FFDC Error Stack file could not be locked for exclusive use by this interface. Repeated attempts had been made to lock this file, and all attempts failed. Another process may have locked the file and failed to release it, or the other process may be hung and is preventing other processes from using the FFDC Error Stack. No FFDC Failure Identifier is provided by this command.

24 No information recorded to the FFDC Error Stack, and no FFDC Failure Identifier is provided by this command. The FFDC Error Stack file appears to be corrupted. The client should consider the FFDC Error Stack Environment unusable.

25 No information recorded to the FFDC Error Stack, and no FFDC Failure Identifier is provided by this command. The FFDC Error Stack file name is set to a directory name. The FFDC Error Stack Environment should be considered corrupted and unusable.

32 A dump file could not be copied to the **/var/adm/ffdc/dumps** directory. There is insufficient space

in the file system containing the **/var/adm/ffdc** directory. The **fcclear** command should be used to remove unneeded FFDC Error Stacks and dump files, or the system administrator needs to add more space to the file system. No FFDC Failure Identifier is provided by this command.

40 No information recorded to the FFDC Error Stack - information could not be recorded in the FFDC Error Stack. There is insufficient space in the file system containing the **/var/adm/ffdc** directory. The **fcclear** command should be used to remove unneeded FFDC Error Stacks and dump files, or the system administrator needs to add more space to the file system. No FFDC Failure Identifier is provided by this command.

41 No information recorded to the FFDC Error Stack, and no FFDC Failure Identifier is provided by this command. A failure occurred when reading control information from the FFDC Error Stack or writing incident information to the FFDC Error Stack. The client should conclude that the entry was not recorded for this incident.

99 No information recorded to the FFDC Error Stack, and no FFDC Failure Identifier is provided by this command. An unexpected internal failure occurred in the **fc_push_stack** routine.This problem may require the attention of application-support services.

When **fcpushstk** is provided with incomplete information, it substitutes default information for the missing information and attempts to make a record in the FFDC Error Stack. Warnings are generated in these cases, and warning messages are displayed to the standard error device unless the **-q** option has been specified. In cases where more than one warning condition was detected, the command generates an exit status code corresponding to the most severe warning condition it detected. The following exit status codes are returned by **fcpushstk** when warning conditions are detected:

26 Both a detailed data string and a detail data file were provided to this routine. The routine chose the detail data string and ignored the detail data file.

28 The name of the resource detecting the incident was not provided. The default resource name was substituted for the missing resource name.

29 At least one component of the detecting application information—source code file name, source code file version, LPP name, line of code position—was not provided. Default information was substituted for the missing information.

30 No default message was provided to describe the nature of the incident. If the XPG/4 message catalog containing the description message cannot be found, no description for this condition will be displayed by the **fcstkrpt** command.

31 No message was provided to describe the nature of the incident, or a component of the XPG/4 information—catalog file name, message set number, message number—was not provided. No description for this condition can be displayed by the **fcstkrpt** command.

32 The file named in the *detail_data_file* parameter could not be copied to the **/var/adm/ffdc/dumps** directory. The FFDC Error Stack entry cites the original version of this file. Do not discard the original copy of this file.

35 No detailed information was provided for this incident. Later problem analysis may be difficult without these details to indicate specifics on the incident.

37 An FFDC Failure Identifier could not be constructed for the report created by this routine. No FFDC Failure Identifier is provided by this command, but information on the incident was recorded to the FFDC Error Stack.

44 The information provided to this command would have caused an FFDC Error Stack record to exceed the FC_STACK_MAX limit. The record was truncated to allow it to be recorded within the system limits. Important information about the failure may have been lost during the truncation process. Modify the script to provide less information, or to record the information to a detail data file and submit the detail data file name to this command instead.

Examples

To record information about a failure to the FFDC Error Stack when the FFDC Environment is established or inherited by the process:

```
#!/bin/ksh
    :
    :
cp /tmp/workfile $FILENAME
RC=$?
if ((RC != 0))
then
    FFDCID=$(fcpushstk -c mymsg.cat -m2 -n10 -o$FILENAME -r myprog
         -d"cp exit status $RC - file being copied /tmp/workfile" -s$0
         -p$LINENO -v"1.1" -lPSSP "Cannot update configuration file %1$s")
    if (($? == 0))
    then
         fcdispfid $FFDCID
         return 1
    fi
fi
    :
    :
```

To make the same recording from a script language that does not have a line of code variable available:

```
#!/bin/bsh
    :
    :
CODESCTN=14             # Used to identify where in the script code we are
cp /tmp/workfile $FILENAME
RC=$?
if test $RC -ne 0
then
    FFDCID=`fcpushstk -c mymsg.cat -m2 -n10 -o$FILENAME -r myprog
         -d"cp exit status $RC - file being copied /tmp/workfile" -s$0
         -p$CODESCTN -v"1.1" -lPSSP "Cannot update configuration file %1$s"`
    if test $? -eq 0
    then
         fcdispfid $FFDCID
         return 1
    fi
fi
CODESECTION=15              # New code section begins - a different task starts
    :
    :
```

To record information about a failure condition that is related to another failure condition previously recorded to the FFDC Error Stack by an application exploiting FFDC:

```
#!/bin/ksh
    :
    :
ASSOC_FID=$(/usr/lpp/ssp/bin/somecmd -a -b)
RC=$?if ((RC != 0))
then
    FFDCID=$(fcpushstk -a$ASSOC_FID -c mymsg.cat -m2 -n10 -o$FILENAME -r myprog
         -d"cp exit status $RC - file being copied /tmp/workfile" -s$0
         -p$LINENO -v"1.1" -lPSSP "Cannot update configuration file %1$s")
    if (($? == 0))
    then
         fcdispfid $FFDCID
         return 1
```

```
    fi
fi
    :
    :
```

Related Information

Commands: **fcdecode**, **fcdispfid**, **fcinit**, **fcreport**, **fcstkrpt**, **fcteststk**

Subroutines: **fc_push_stack** (see the *RSCT First Failure Data Capture Programming Guide and Reference*)

---

fcreport Command

Purpose

Locates and displays the report of a failure and any failures associated with the failure.

Syntax

/usr/sbin/rsct/bin/fcreport { [**-a**] *FFDC_Failure_ID* } | **-h**

Description

fcreport decodes an FFDC Failure Identifier, and obtains reports on the failure identified by it. The command also detects if any failure was associated with the FFDC Failure Identifier, and if so, obtains the report on that failure. The command continues to examine the report of each failure it locates for associated failures and to obtain reports on the associated failures until one of the following conditions is met:

- No further associated failures are detected.
- The report for an associated failure cannot be found. This may occur when the associated failure report resides on a remote node that cannot be reached at the moment, or the record of the failure has been removed from the node where it resided.

Using this command, the user can obtain a report for the entire list of failures that caused a specific failure. **fcreport** is not capable of locating reports for any failures that may have been caused by the initial failure provided to the command; it can only obtain reports of failures that caused this failure.

Flags

-a Displays all information contained in a report for a failure. The default is to display the network address of the node where the failure report was generated, the time stamp on the failure report, and the description of the incident recorded in the failure report.

-h Displays a help message to standard output and exits. No other processing is performed, regardless of the options specified.

Parameters

FFDC_Failure_ID

Specifies the FFDC Failure Identifier of the failure to begin the report. **fcreport** will attempt to obtain the failure information for this failure, as well as any failures that this report lists as an associated failure. Only one FFDC Failure Identifier may be provided to this command.

Security

fcreport uses **rsh** to obtain failure reports that may reside on remote nodes. The user must have sufficient privilege to execute **rsh** commands to these remote nodes. If the user does not have this permission, **fcreport** can only trace the list of related failures so long as they exist on the local node.

Exit Status

fcreport generates one of the following exit status codes upon completion:

0 Failure report located and displayed for the FFDC Failure Identifier provided. Zero or more related failure reports may have been located and displayed as well.

2 Help information displayed and processing ended.

10 Required options or arguments are not provided.

11 The FFDC Failure Identifier provided to this command was generated by a later release of the FFDC software. The command is not capable of correctly interpreting this identifier.

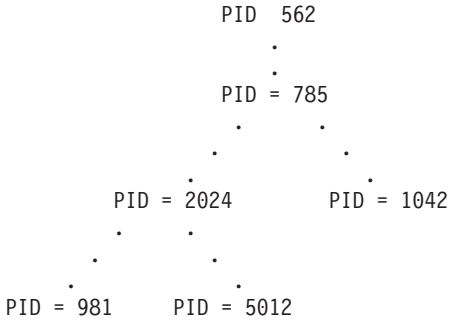
12 Unknown option specified to this command.

20 The FFDC Failure Identifier refers to an entry made in an FFDC Error Stack on this system, but the FFDC Error Stack file cannot be accessed. The file may have been removed, or permissions may have been altered on the file to prevent access to it.

27 The FFDC Failure Identifier provided to this command is not a valid identifier.

Examples

Consider the case where several processes were created in the following parent-child order:



In this example, process 785 generated the FFDC Failure Identifier **.3Iv04ZVVfvp.wtY0xRXQ7.....** and passed it back to Process 562. To obtain a detailed report for FFDC Failure Identifier **.3Iv04ZVVfvp.wtY0xRXQ7.....** and any previous failures that led to this specific failure:

```
$ fcreport -a .3Iv04ZVVfvp.wtY0xRXQ7...................
```

This report will contain the details of the specified FFDC Failure Identifier, as well as any failures in processes 2024, 1042, 981, and 5012 that may have caused it. The report will not contain any failures in process 562 that may have been caused as a result of process 785's failure.

Related Information

Commands: **fcclear**, **fcdecode**, **fcdispfid**, **fcfilter**, **fclogerr**, **fcpushstk**, **fcstkrpt**

Subroutines: **fc_log_error**, **fc_push_stack** (see the *RSCT First Failure Data Capture Programming Guide and Reference*)

---

fcstat Command

Purpose

Displays statistics gathered by the specified Fibre Channel device driver.

Syntax

fcstat *Device_Name*

Description

The **fcstat** command displays the statistics gathered by the specified Fibre Channel device driver. It collects the statistics using the following procedure:

1. Opens the message catalog of **fcstat** and checks the parameter list.
2. Accesses the ODM database for information relating to the selected adapter.
3. Accesses the ODM database for information relating to ports of the selected adapter.
4. Opens and accesses adapter statistics.
5. Reports statistics and exits.

If an invalid *Device_Name* is specified, the **fcstat** command returns an error message stating that it could not find the device in the ODM database. The **fcstat** command also reports an error if the specified *Device_Name* is not connected to a network (that is, the link is down). When the **fcstat** command is not able to extract statistics from the specified *Device_Name*, it still reports the information that it extracted from the ODM database.

Parameters

Device_Name	The name of the Fibre Channel device. For example, `fcs0`.

Statistics Fields

Note: Some adapters might not support a specific statistic. The value of nonsupported statistic fields is always 0.

The statistic fields displayed in the output of the **fcstat** command and their descriptions are:

`Device Type`	Displays the description of the adapter.
`Serial Number`	Displays the serial number from the adapter.
`Option ROM Version`	Displays the version of the Options ROM on the adapter.
`Firmware Version`	Displays the version of the firmware on the adapter.
`Node WWN`	Displays the worldwide name of the adapter.
`Port FC ID`	Displays the SCSI ID of the adapter.
`Port Type`	Displays the adapter's connection type.
`Port Speed`	Displays the speed of the adapter.
`Port WWN`	Displays the worldwide name of the port.
`Seconds Since Last Reset`	Displays the seconds since last reset of the statistics on the adapter.
`Frames`	Displays the number of frames transmitted and received.
`Words`	Displays the number of words transmitted and received.
`LIP Count`	Displays the LIP count.
`NOS Count`	Displays the NOS count.
`Error Frames`	Displays the number of frames that were in error.
`Dumped Frames`	Displays the frames that were dumped.
`Link Failure Count`	Displays the Link Failure Count.
`Loss of Sync Count`	Displays the number of times Sync was lost.
`Loss of Signal`	Displays the number of times signal was lost.
`Primitive Seq Protocol Err Count`	Displays the number of times a primitive sequence was in error.
`Invalid Tx Word Count`	Displays the number of invalid transfers that occurred.
`Invalid CRC Count`	Displays the number of CRC errors that occurred.

IP over FC Adapter Driver Information: No DMA Resource Count	Displays the number of times DMA resources were not available.
IP over FC Adapter Driver Information: No Adapter Elements Count	Displays the number of times there were no adapter elements available.
FC SCSI Adapter Driver Information: No DMA Resource Count	Displays the number of times DMA resources were not available.
FC SCSI Adapter Driver Information: No Adapter Elements Count	Displays the number of times there were no adapter elements available.
FC SCSI Adapter Driver Information: No Command Resource Count	Displays the number of times there were no command resources available.
IP over FC Traffic Statistics: Input Requests	Displays the number of input requests.
IP over FC Traffic Statistics: Output Requests	Displays the number of output requests.
IP over FC Traffic Statistics: Control Requests	Displays the number of control requests.
IP over FC Traffic Statistics: Input Bytes	Displays the number of input bytes.
IP over FC Traffic Statistics: Output Bytes	Displays the number of output bytes.
FC SCSI Traffic Statistics: Input Requests	Displays the number of input requests.
FC SCSI Traffic Statistics: Output Requests	Displays the number of output requests.
FC SCSI Traffic Statistics: Control Requests	Displays the number of control requests.
FC SCSI Traffic Statistics: Input Bytes	Displays the number of input bytes.
FC SCSI Traffic Statistics: Output Bytes	Displays the number of output bytes.

Exit Status

0	The command completed successfully.
>0	An error occurred.

Examples

1. To display the statistics for Fiber Channel device driver `fcs0`, enter:

   ```
   fcstat fcs0
   ```

   Output similar to the following is displayed.

   Note: The output format of various AIX commands is not always static. Do not write programs with the expectation that the output for the **fcstat** command will remain as follows.

```
FIBRE CHANNEL STATISTICS REPORT: fcs0

    Device Type: FC Adapter (df1000f9)
    Serial Number: 1E313BB001
    Option ROM Version: 02C82115
    Firmware Version: B1F2.10A5
    Node WWN: 20000000C9487B04
    Port WWN: 10000000C9416DA4

    FC4 Types
      Supported: 0x0000010000000000000000000000000000000000000000000000000000000000
      Active:    0x0000010000000000000000000000000000000000000000000000000000000000
    Class of Service: 4
    Port FC ID: 011400
    Port Speed (supported): 2 GBIT
    Port Speed (running):   1 GBIT
    Port Type: Fabric

    Seconds Since Last Reset: 345422

    Transmit Statistics    Receive Statistics
    -------------------    ------------------
    Frames: 1              Frames: 1
```

```
Words: 1               Words: 1

LIP Count: 1
NOS Count: 1
Error Frames:  1
Dumped Frames: 1
Link Failure Count: 1
Loss of Sync Count: 1
Loss of Signal: 1
Primitive Seq Protocol Err Count: 1
Invalid Tx Word Count: 1
Invalid CRC Count: 1

IP over FC Adapter Driver Information
  No DMA Resource Count: 0
  No Adapter Elements Count: 0

FC SCSI Adapter Driver Information
  No DMA Resource Count: 0
  No Adapter Elements Count: 0
  No Command Resource Count: 0

IP over FC Traffic Statistics
  Input Requests:   0
  Output Requests:  0
  Control Requests: 0
  Input Bytes:  0
  Output Bytes: 0

FC SCSI Traffic Statistics
  Input Requests:   16289
  Output Requests:  48930
  Control Requests: 11791
  Input Bytes:  128349517
  Output Bytes: 209883136
```

Location

/usr/sbin/fcstat

Related Information

The **atmstat** command, “entstat Command” on page 358, **fddistat** command, **netstat** command, **tokstat** command.

---

fcstkrpt Command

Purpose

Displays the contents of an FFDC Error Stack file.

Syntax

/usr/sbin/rsct/bin/fcstkrpt { [**-a**] [**-p** | **-r**] { **-f** *FFDC_Failure_Identifier* [**-i**] | **-s** *FFDC_Error_Stack_File_Name* } } | [**-h**]

Description

fcstkrpt reads an existing FFDC Error Stack file and displays its contents to the standard output device. The FFDC Error Stack file is indicated either through the name of the file itself, or by using an FFDC Failure Identifier that references a specific record within that file.

Information from the FFDC Error Stack can be displayed in one of two formats: by *related failure conditions* (the default) or by *software layer*.

Flags

-a Indicates that all information be displayed for entries in the FFDC Error Stack. The default action is to display the time stamp for the record and the description of the incident.

-f Specifies the FFDC Failure Identifier to use in locating the FFDC Error Stack. **fcstkrpt** decodes the FFDC Failure Identifier, locates the FFDC Error Stack associated with that FFDC Failure Identifier, and processes the FFDC Error Stack. Only one FFDC Failure Identifier can be specified by this flag.

-h Displays a help message to standard output and exits. No other processing is performed regardless of the options specified.

-i Displays only the information associated with the specific failure report identified by the **-f** flag. By default, all records in the FFDC Error Stack are displayed.

-p Displays information from the FFDC Error Stack by process orientation. The output is ordered so that it reflects the order in which the processes were created (parent-child process relationship). Child process information is shown first, followed by parent process information. This view is used to understand which incidents occurred first, and which incidents occurred later because of them.

-r Displays information from the FFDC Error Stack by incident relationships. Incidents are presented along with those incidents that are related to them. This view is used to understand which incidents occurred because of the occurrence of other incidents. This is the default.

-s Specifies the name of the FFDC Error Stack to be examined. This name may be either the absolute or relative path name of the FFDC Error Stack. Only one FFDC Error Stack file name can be specified by this flag. If a relative file name is used, the file is assumed to be located in the **/var/adm/ffdc/stacks** directory of the node where the file resides.

Parameters

FFDC_Failure_ID

Specifies the FFDC Failure Identifier of the failure to begin the report. **fcreport** will attempt to obtain the failure information for this failure, as well as any failures that this report lists as an associated failure. Only one FFDC Failure Identifier may be provided to this command.

Security

fcreport uses **rsh** to obtain failure reports that may reside on remote nodes. The user must have sufficient privilege to execute **rsh** commands to these remote nodes. If the user does not have this permission, **fcreport** can only trace the list of related failures so long as they exist on the local node.

Exit Status

fcstkrpt issues the following integer exit status codes upon completion:

0 FFDC Error Stack file successfully located, and contents displayed to the standard output device.

2 Help information displayed and processing ended.

12 An invalid option was specified.

14 No information written to the standard output device. The **-f** option was used and the *FFDC Error Identifier* argument was not valid.

20 No information written to the standard output device. The **-s** option was used and the *FFDC Error Stack File* argument was not found.

27 No information written to the standard output device. The caller provided a valid *FFDC Failure Identifier*, but the file referenced by the FFDC Failure Identifier was not recorded on this node. Use the **fcdecode** command to locate the node where this FFDC Error Stack resides.

81 No information written to the standard output device. A failure occurred while writing information to standard output. The application should conclude that standard output cannot accept output.

85 No information written to the standard output device. The caller provided a valid FFDC Failure Identifier, but the file referenced by the FFDC Failure Identifier does not exist.

Examples

To obtain a brief report of the information stored in the FFDC Error Stack file **/var/adm/ffdc/stacks/myprog.562.19981001143052**:

```
$ fcstkrpt -r -s myprog.562.19981001143052
```

To obtain a detailed report of the information contained in the FFDC Error Stack where the FFDC Failure Identifier **.3Iv04ZVVfvp.wtY0xRXQ7.....** was recorded, and present this information in parent-child ordering:

```
$ fcstkrpt -p -f .3Iv04ZVVfvp.wtY0xRXQ7...................
```

Related Information

Commands: **fcclear**, **fcdecode**, **fcdispfid**, **fcfilter**, **fcpushstk**, **fcreport**

Subroutines: **fc_push_stack** (see the *RSCT First Failure Data Capture Programming Guide and Reference*)

---

fcteststk Command

Purpose

Test for the presence of a First Failure Data Capture Error Stack environment.

Syntax

/usr/sbin/rsct/bin/fcteststk [**-q**] | [**-h**]

Description

fcteststk can be called by any application program that wishes to use the FFDC Error Stack to test if these facilities have been activated . By performing this test, applications can avoid the performance burden of collecting failure information in cases where an *FFDC Environment* has not been established. This interface is provided primarily for use by library routines, which would not have any knowledge of whether their client application established or inherited an *FFDC Environment*.

An *FFDC Error Stack Environment* is established by a process when that process wants to have failure information from itself, any threads it may create, and any descendant processes it may create to be recorded in an FFDC Error Stack. An *FFDC Error Stack Environment* is inherited by a process when that process wants to record failure information to an FFDC Error Stack file only when one of its ancestors has requested for processes to do so; in all other cases, the process will not record failure information to the FFDC Error Stack. Processes use **fcinit** to either establish or inherit the FFDC Error Stack Environment.

The FFDC Error Stack Environment reserves an FFDC Error Stack file, so that failure information is recorded to a file in the **/var/adm/ffdc/stacks** directory. These files use the naming format ***script_name.PID.date_and_time***, where *script_name* is the name of the script itself, *PID* is the process identifier of the script, and *date_and_time* is the date and time when the script was executed. Whenever this script or children processes of this script record failure information to the FFDC Error Stack, it will be recorded in this file.

Applications use the **fcpushstk** interface to record failure information to the FFDC Error Stack. However, the application may need to collect this information from various locations before recording the information, and obtaining this information can impact the application's overall performance. The application should not need to collect this information if the *FFDC Error Stack Environment* was not established or inherited. To avoid this performance impact, the application can issue **fcteststk** to determine if an *FFDC Error Stack Environment* is available, and if so, begin collecting the failure information. If the *FFDC Error Stack Environment* does not exist, the application can avoid collecting this information.

Processes that use the **fclogerr** FFDC interface can use **fclogerr** when an *FFDC Environment* exists, whether or not an FFDC Error Stack is in use by the *FFDC Environment*. Whenever **fclogerr** is used, failure information is recorded to the AIX Error Log and the BSD System Log, regardless of whether an FFDC Error Stack was reserved. Any application that records information using the **fclogerr** interface must *always* collect the failure information and record it, regardless of whether an FFDC Error Stack is in use.

Flags

-h Displays a usage message for this command. No further processing is performed.

-q Suppresses output from this command that explains whether or not an FFDC Environment was established. The command user will be required to test the exit status from the command to determine whether an FFDC Environment is established for this process.

Parameters

FFDC_Failure_ID

Specifies the FFDC Failure Identifier of the failure to begin the report. **fcreport** will attempt to obtain the failure information for this failure, as well as any failures that this report lists as an associated failure. Only one FFDC Failure Identifier may be provided to this command.

Security

fcreport uses **rsh** to obtain failure reports that may reside on remote nodes. The user must have sufficient privilege to execute **rsh** commands to these remote nodes. If the user does not have this permission, **fcreport** can only trace the list of related failures so long as they exist on the local node.

Exit Status

0 An FFDC Error Stack Environment exists.

2 Help information displayed and processing ended.

12 No processing performed. An invalid option was specified.

15 FFDC Error Stack Environment has not been established or inherited by the client at this point in time.

17 FFDC Error Stack Environment appears to be corrupted and should be considered unusable.

Examples

To test whether an FFDC Error Stack Environment exists for an application:

```
fcteststk -q
if (($? == 0))
then
    # Collect failure information
        :
        :
```

```
        # Use fcpushstk to record failure info
             :
             :
    fi
```

Related Information

Commands: **fcinit**, **fcpushstk**

Subroutines: **fc_test_stack** (see the *RSCT First Failure Data Capture Programming Guide and Reference*)

---

fddistat Command

Purpose

Shows FDDI device driver and device statistics.

Syntax

fddistat [**-r -t**] *Device_Name*

Description

The **fddistat** command displays the statistics gathered by the specified FDDI device driver. If no flags are specified, only the device driver statistics are displayed. This command is also invoked when the **netstat** command is run with the **-v** flag. The **netstat** command does not issue any **fddistat** command flags.

If an invalid *Device_Name* is specified, the **fddistat** command will produce an error message stating that it could not connect to the device.

Flags

-r Resets all the statistics back to their initial values. This flag can only be issued by privileged users.
-t Toggles debug trace in some device drivers.

Parameter

Device_Name The name of the FDDI device, for example, **fddi0**.

Statistic Fields

Note: Some adapters may not support a specific statistic. The value of non-supported statistic fields is always 0.

The statistic fields displayed in the output of the **fddistat** command and their descriptions are:

Title Fields

`Elapsed Time` Displays the real time period has elapsed since last time the statistics was reset. Since part of the statistics may be reset by the device driver during error recovery when a hardware error was detected, there will be another Elapsed Time displayed in the middle of the output when this situation has occurred in order to reflect the time differences between the statistics.

Transmit Statistics Fields

`Packets` The number of packets transmitted successfully by the device.

Bytes	The number of bytes transmitted successfully by the device.
Interrupt	The number of transmit interrupts received by the driver from the adapter.
Transmit Errors	The number of output errors encountered on this device. This is a counter for unsuccessful transmissions due to hardware/network errors.
Packets Dropped	The number of packets accepted by the device driver for transmission which were not (for any reason) given to the device.
Max Packets on S/W Transmit Queue	The maximum number of outgoing packets ever queued to the software transmit queue.
S/W Transmit Queue Overflow	The number of outgoing packets overflowed the software transmit queue.
Current S/W+H/W Transmit Queue Length	The number of pending outgoing packets on either the software transmit queue or the hardware transmit queue.
Broadcast Packets	The number of broadcast packets has been transmitted without any error.
Multicast Packets	The number of multicast packets has been transmitted without any error.

Receive Statistics Fields

Packets	The number of packets has been received successfully by the device.
Bytes	The number of bytes received successfully by the device.
Interrupts	The number of receive interrupts received by the driver from the adapter.
Receive Errors	The number of input errors encounteredon this device. This is a counter for unsuccessful reception due to hardware/network errors.
Packets Dropped	The number of packets received by the device driver from this device which were not (for any reason) given to a network demuxer.
Bad Packets	The number of bad packets received (i.e.saved) by the device driver.
Broadcast Packets	The number of broadcast packets received without any error.
Multicast Packets	The number of multicast packets received without any error.

General Statistics Fields

No mbuf Errors	The number of times that mbufs were not available to the device driver. This usually occurs during receive operations when the driver must obtain mbuf buffers to process inbound packets. If the mbuf pool for the requested size is empty, the packet will be discarded. The **netstat -m** command can be used to confirm this.
SMT Error Word	The adapter's SMT error status.
SMT Event Word	The adapter's SMT event status.
Connection Policy Violation	The status of the adapter's connection to the ring.
Port Event	The adapter's port status.
Set Count	The current set count value.
Adapter Check Code	The adapter's most recent adapter check status.
Purged Frames	Receive frames dropped by the adapter due to lack of available descriptors.
ECM State Machine	Entity Coordination Management State Machine.
PCM State Machine: Port A	Physical Connection Management for the primary adapter State Machine
PCM State Machine: Port B	Physical Connection Management for the secondary adapter State Machine

CFM State Machine: Port A	Configuration Management for the primary adapter State Machine
CFM State Machine: Port B	Configuration Management for the secondary adapter State Machine
CF State Machine	Overall Configuration State Machine.
MAC CFM State Machine	Configuration Management for the MAC State Machine.
RMT State Machine	Ring Management State Machine.
Driver Flags	The device driver internal status flags that are currently turned on.

Example

To display the device driver statistics for **fddi0**, enter:

```
fddistat fddi0
```

This produces the following output:

```
------------------------------------------------------------
FDDI STATISTICS (fddi0) :
Elapsed Time: 0 days 0 hours 1 minutes 3 seconds

Transmit Statistics:                          Receive Statistics:
--------------------                          -------------------
Packets: 100                                  Packets: 100
Bytes: 113800                                 Bytes: 104700
Interrupts: 100                               Interrupts: 100
Transmit Errors: 0                            Receive Errors: 0
Packets Dropped: 0                            Packets Dropped: 0
Max Packets on S/W Transmit Queue: 0          Bad Packets: 0
S/W Transmit Queue Overflow: 0
Current S/W+H/W Transmit Queue Length: 0

Broadcast Packets: 0                          Broadcast Packets: 0
Multicast Packets: 0                          Multicast Packets: 0

General Statistics:
-------------------
No mbuf Errors: 0
SMT Error Word: 00040080                      SMT Event Word: 000004a0
Connection Policy Violation: 0000             Port Event: 0000
Set Count Hi: 0000                            Set Count Lo: 0003
Adapter Check Code: 0000                      Purged Frames: 0

ECM State Machine:        IN
PCM State Machine Port A: CONNECT
PCM State Machine Port B: ACTIVE
CFM State Machine Port A: ISOLATED
CFM State Machine Port B: CONCATENATED
CF State Machine:         C_WRAP_B
MAC CFM State Machine:    PRIMARY
RMT State Machine:        RING_OP

Driver Flags: Up Broadcast Running
        Simplex DualAttachStation
```

Related Information

The **atmstat** command, **entstat** command, **netstat** command, **tokstat** command.

fdformat Command

Purpose

The **fdformat** command formats diskettes.

Syntax

fdformat [*Device*] [**-h**]

Description

Attention: Formatting a diskette or read/write optical disk destroys any existing data on it.

The **fdformat** command formats diskettes in the diskette drive specified for low density unless the **-h** flag is specified.

All new, blank diskettes must be formatted before they can be used.

Before formatting a diskette or read/write optical disk, the **fdformat** command prompts for verification. This allows you to end the operation cleanly.

Flags

-h Forces high-density formatting. This flag is used only with the **fdformat** command.

Parameters

Device Specifies the device containing the diskette to be formatted. The default is the **/dev/rfd0** device for drive 0.

Examples

To force high-density formatting of a diskette when using the **fdformat** command, enter:

```
fdformat -h
```

Files

/usr/sbin/fdformat	Contains the **fdformat** command.
/dev/rfd*	Specifies the device parameters.
/dev/fd*	Specifies the device parameters.
/dev/romd*	Specifies the device parameters.
/dev/omd*	Specifies the device parameters.

Related Information

The **flcopy** command, **format** command.

The **fd** special file.

---

fdpr Command

Purpose

A performance tuning utility for improving execution time and real memory utilization of user-level post-link application programs.

Syntax

Most Common Usage:

fdpr -p *ProgramFile* **-x** *WorkloadCommand*

Detailed Usage:

fdpr -p *ProgramFile* [**-M** *Segnum*] [**-fd** *Fdesc*] [**-o** *OutputFile*] [**-armember** *ArchiveMemberList*] [OptimizationFlags] [**-map**] [**-disasm**] [**-disasm_data**] [**-disasm_bss**] [**-profcount**] [**-quiet**] [**-v**] [**-1 | -2** | **-3 | -12** | **-23 | -123**] [**-x** *WorkloadCommand*]

Optimization Flags

[**-tb**] [**-pc**] [**-pp**] [**-O**][**-O2**] [**-O3**] [**-O4**] [**-selective_inline**] [**-sid_fac** *percent*] [**-inline_small_funcs** *size*] [**-inline_hot_funcs** *percent*] [**-hco_resched**] [**-killed_regs**] [**-lr_opt**] [**-align** *bytes*] [**-RD**] [**-dpnf** *factor*] [**-dpht** *threshold*] [**-build_dcg**] [**-tocload**] [**-ptrgl_opt**] [**-no_ptrgl_r11**] [**-dcbt_opt**] [**-ignore_info**] [**-dead_code_removal**] [**-bt_csect_anchor_removal**] [**-strip**] [**-analyse_asm_csects**] [**-extra_safe_analysis**] [**-inline**] [**-reduce_toc** *removal_factor*]

Description

The **fdpr** command (Feedback Directed Program Restructuring) is a performance-tuning utility that may help improve the execution time and the real memory utilization of user-level application programs. The **fdpr** program optimizes the executable image of a program by collecting information on the behavior of the program while the program is used for some typical workload, and then creating a new version of the program that is optimized for that workload. The new program generated by **fdpr** typically runs faster and uses less real memory.

Attention: The **fdpr** command applies advanced optimization techniques to a program which may result in programs that do not behave as expected; programs which are optimized using this tool should be used with due caution and should be rigorously retested with, at a minimum, the same test suite used to test the original program in order to verify expected functionality. The optimized program is not supported.

The **fdpr** command builds an optimized executable program in 3 distinct phases:

- Phase 1 (**-1** flag): Creates an instrumented executable program and an empty template profile file.
- Phase 2 (**-2** flag): Runs the instrumented program and updates the profile data.
- Phase 3 (**-3** flag): Generates the optimized executable program file.

These phases can be run separately or in partial or full combination, but must be run in order (i.e., **-1** then **-2** then **-3** or **-12** then **-3**). The default is to run all three phases.

Note: The instrumented executable, created in phase 1 and run in phase 2, typically runs several times slower than the original program. Due to the increased execution time required by the instrumented program, the executable should be invoked in such a way as to minimize execution duration, while still fully exercising the desired code areas. The **fdpr** command user should also attempt to eliminate, where feasible, any time dependent aspects of the program.

Flags

-1,**-2**, **-3**	Specifies the phase to run. The default is all 3 phases (**-123**). The **-s** flag must be used when running separate phases so that the succeeding phases can access the required intermediate files. The phases must be run in order (for example, **-1**, then **-2**, then **-3**, or **-1**, then **-23**). The **-2** flag must be used along with the invocation flag **-x**.
-M *SegNum*	Specifies where to map shared memory for profiling. The default is **0x30000000**. Specify an alternate shared memory address if the program to be optimized or any of the workload command strings invoked with the **-x** flag use conflicting shared-memory addresses. Typical alternative values are **0x40000000**, **0x50000000**, ... up to **0xC0000000**).
-fd *Fdesc*	Specifies which file descriptor number is to be used for the **profile** file that is mapped to the above shared memory area. The default of *Fdesc* is set to 1999.
-o *OutFile*	Specifies the name of the output file from the optimizer. The default is *program***.fdpr**
-p *ProgramFile*	Contains the name of the executable program file or shared object file or shared library containing shared objects/executables, to optimize. This program must be an unstripped executable.

-armember *ArchiveMemberList*	List of archive members to be optimized, within a shared archive file specified by the **-p** flag. If **-armember** is not specified, all members of the archive file are optimized.
-map	Print a map of basic blocks and static variables with their respective old -> new addresses into a suffixed **.mapper** file.
-disasm	Prints the disassembled text section of the output optimized and instrumented program into a suffixed **.dis_text** file.
-disasm_data	Prints the disassembled data section of the output optimized and instrumented program into a suffixed **.dis_data** file.
-disasm_bss	Prints the disassembled bss section of the output optimized and instrumented program into a suffixed **.dis_bss** file.
-profcount	Prints the profiling counters into a suffixed **.ncounts** file.
-quiet	Quiet output mode.
-v	Verbose output.
-x *WorkloadCommand*	Specifies the command used for invoking the instrumented program. All the arguments after the **-x** flag are used for the invocation. Therefore, the **-x** flag must appear last in the command line. The **-x** flag is required when the **-2** flag is used.

Optimization Flags

-analyse_asm_csects	Analyze csects written in assembly (when used, must be specified at both the -1 and -3 phases).
-extra_safe_analysis	Do not attempt to analyze unconventional csects containing hand-written assembly code (when used, must be specified at both the -1 and -3 phases).
-ignore_info	Ignore **.info** sections produced with the **-qfdpr** option during compile time (when used, must be specified at both -1 and -3 phases).
-align *bytes*	Align frequently executed code according to given number of bytes, for improving code prefetch buffer ratio. If this option is omitted, the **fdpr** command aligns the code with variable default number of bytes.
-lr_opt	Eliminate stores and restores of the link register in frequently executed procedures.
-bt_csect_anchor_removal	Eliminate load instructions related to the usage of branch tables in the code.
-dead_code_removal	Remove unreachable code.
-selective_inline	Perform selective inlining for functions that are frequently called from a single dominant call site.
-sid_fac *percent*	Set a dominant factor percentage for selective inline optimization. The allowed range is between 50 - 100 (applicable only with the **-selective_inline** flag).
-inline_small_funcs *size*	Inline all functions that are smaller or equal to the given size in bytes.
-inline_hot_funcs *percent*	Inline all functions with an execution frequency equals or greater than the given percentage. The input percent range is between 0 - 100.
-inline	Perform **-inline_small_funcs 12** with **-selective_inline**.
-hco_resched	Relocate instructions from frequently executed code to rarely executed code area, when possible.
-dcbt_opt	Insert **dcbt** instructions to improve data-cache performance.
-killed_regs	Eliminate stores and restores of registers that are *killed* (overwritten) after frequently executed function calls.
-tb	Force the restructuring of traceback tables in reordered code. If **-tb** option is omitted, traceback tables are automatically restored for C++ applications using Try & Catch mechanism.
-pc	Preserve csects' boundaries in reordered code.
-pp	Preserve functions' boundaries in reordered code.
-RD	Perform static data reordering.
-dpnf *factor*	Data Placement Normalization Factor between 0 - 1; where 0 causes static variables to be reordered regardless of their size, whereas 1 will locate only small sized variables first (applicable only with the **-RD** flag).

-dpht *threshold*	Data Placement Hotness Threshold between 0 - 1; where 0 reorders the static variables in large groups based on the control flow, and whereas 1 will reorder the variables in very small groups based on their access frequency (applicable only with the **-RD** flag).
-build_dcg	Build DCG (Data Connectivity Graph) for enhanced data reordering (applicable only with the **-RD** flag).
-tocload	Perform tocload optimization.
-reduce_toc *removal_factor*	Perform TOC entries removal accordingly to removal factor between 0 - 1, where 0 removes only non-accessed TOC entries and 1 removes all non-exported TOC entries.
-strip	Strip the output file (if any is produced).
-ptrgl_opt	Perform optimization of indirect call instructions by way of registers by replacing them with direct jumps.
-no_ptrgl_r11	Do not perform removal of R11 load instruction in _ptrgl csect (the **-ptrgl_r11** optimization is applied by default).
-O	Perform code reordering with branch prediction bit setting, branch folding and NOOP instructions removal. The **-O** flag is applied by default.
-O2	Switch on all less aggressive optimization flags.
-O3	Switch on all aggressive optimization flags.
-O4	Switch on all aggressive optimization flags.

Optimization

The **fdpr** command performs, by default, the highest possible level of code reordering optimization together with the optimizations of branch prediction bit setting, branch folding, code alignment and removal of redundant NOOP instructions. The **-pc** flag reorders the entire code while preserving csects' boundaries and therefore, may result in less performance improvement than the default code reordering. Similarly, the **-pp** flag reorders the entire code while preserving procedures' boundaries.

Additional optimizations performed on the entire executable program file are available by the optimization flags above.

Executables built with the **-qfdpr** IBM xl compiler flag contain information to assist **fdpr** in producing reordered programs. Modules which are not compiled with the **-qfdpr** option, are reordered based on the compiler signatures in the symbol table.

Additional performance enhancements may be realized by using static linking when building the program to be reordered. Since the **fdpr** program only reorders the instructions within the executable program specified, any dynamically linked shared library routines called by the program are not optimized. Statically linking these library routines to the executable allows for optimizing both the instructions in the program and all library routines used by the program. There are other advantages as well as disadvantages to building a statically linked program. See the *Performance management* for further information.

Output Files

All files created by the **fdpr** command are stored in the current directory with the exception of any files which may be created by running the workload command specified in the **-x** flag. During the optimization process, the original program is saved by renaming the program, and is only restored to the original program name upon successful completion of the final phase.

The profile file created by the **fdpr** command explicitly uses the full name of the current directory since scripts used to run the program may change the working directory before executing the program.

The files created and/or used by the **fdpr** command are:

program	Name of the unstripped executable to be optimized.
*program***.save**	Saved version of the original executable program.

program.**nprof**	Name of the profile file.
program.**instr**	Name of the instrumented version of program.
program.**fdpr**	Default name of optimized executable output file.
program.**instr.dis_text**	Default disassembly file in ASCII format produced by **-disasm** flag after instrumentation phase.
program.**fdpr.dis_text**	Default disassembly file in ASCII format produced by **-disasm** flag after optimization phase.
program.**instr.dis_data**	Default disassembly file in ASCII format produced by **-disasm_data** flag after instrumentation phase.
program.**fdpr.dis_data**	Default disassembly file in ASCII format produced by **-disasm_data** flag after optimization phase.
program.**instr.dis_bss**	Default disassembly file in ASCII format produced by **-disasm_bss** flag after instrumentation phase.
program.**fdpr.dis_bss**	Default disassembly file in ASCII format produced by **-disasm_bss** flag after optimization phase.
program.**instr.mapper**	Default mapping file in ASCII format produced by **-map** flag after instrumentation phase.
program.**fdpr.mapper**	Default mapping file in ASCII format produced by **-map** flag after optimization phase.
program.**ncounts**	Default profile counters file in ASCII format produced by **-profcount** flag.

Enhanced Debugging Capabilities

In order to enable a certain degree of debugging capability for optimized programs, **FDPR** updates the Symbol Table to reflect the changes that were made in the **.text** section.

Entry fields in the Symbol Table that specify addresses of symbols that were relocated during the reordering of **FDPR**, are modified to point to their new addresses in the **.text** section.

In addition, in the case where functions or files are split during reordering, **FDPR** creates new entries in the Symbol Table for each new part of the split function/file. These new parts of the same function are given new symbol names in the Symbol Table according to the following naming convention:

```
<original function name>__fdpr_<function's part number>
```

After code reordering all the new entries are suffixed with the `__fdpr_` string.

Example: Originally, function ″main″ had the following entry in the Symbol Table:

```
[Index] m   Value       Scn     Aux   Sclass    Type    Name
 [456]  m  0x00000230    2       1     0x02    0x0000   .main
```

If after code reordering, function `main` was split into 3 parts, then it would have 3 entries in the Symbol Table; one for each part as follows:

```
[Index] m   Value       Scn     Aux   Sclass    Type    Name
 [456]  m  0x00000304    2       1     0x02    0x0000   .main
[1447]  m  0x00003328    2       1     0x02    0x0000   .main__fdpr_1
[1453]  m  0x000033b4    2       1     0x02    0x0000   .main__fdpr_2
```

Examples

The following are typical usage examples of the **fdpr** command.

1. This example allows the user to run all three phases. In this example, `test1` is the unstripped executable and `test2` is a shell script that invokes `test1`. The current working directory is **/tmp/fdpr**.

```
test2 script file:

# code to exercise test1
test1 -expand 100 -root $PATH file.jpg -quit
# the end of test2
```

Execute the **fdpr** command (using the default optimization):

```
fdpr -p test1 -x test2
```

This results in the new reordered executable **test1.fdpr**.

2. To run one phase at a time, execute phase one of **fdpr**.

   ```
   fdpr -1 -p test1
   ```

   This command string creates an instrumented version with the name `test1.instr` and the empty template profile file `test1.nprof`.

   To execute phase two:

   ```
   fdpr -2 -p test1 -x test2
   ```

   This command string executes the script file `test2` that runs the instrumented version of `test1` to collect the profile data.

   To execute phase three:

   ```
   fdpr -3 -p test1
   ```

   Again, this results in the new reordered executable **test1.fdpr**.

3. To run the first two phases followed by phase three, execute phase one and two.

   ```
   fdpr -12 -p test1 -x test2
   ```

   Execute phase three using optimization level three.

   ```
   fdpr -3 -O3 -p test1
   ```

4. If an error occurs while running an **fdpr** optimized program, the **dbx** command can be used to determine what procedure the error occurred in as follows:

   ```
   dbx program.fdpr
   ```

   which produces the output similar to the following:

   ```
   Type 'help' for help.
   reading symbolic information ...warning: no source compiled with -g

   [using memory image in core]

   Segmentation fault in proc_d at 0x10000634
   0x10000634 (???) 98640000        stb   r3,0x0(r4)
   (dbx)
   ```

   A stack traceback, which is used to determine how the program arrived at the current location, is produced as follows:

   ```
   (dbx) where
   ```

   which produces the following output:

   ```
   proc_d(0x0) at 0x10000634
   proc_c(0x0) at 0x10000604
   proc_b(0x0) at 0x100005d0
   proc_a(0x0) at 0x1000059c
   main(0x2, 0x2ff7fba4) at 0x1000055c
   (dbx)
   ```

5. The **dbx** subcommand **stepi** may also be used to single step through the instructions of a reordered executable program as follows:

```
(dbx) stepi
```

which produces the following output:

```
stopped in proc_d at 0x1000061c
0x1000061c (???) 9421ffc0       stwu   r1,-64(r1)
(dbx)
```

In this example, **dbx** indicates that the program stopped in routine `proc_d` at address `0x1000061c` in the reordered text section.

Implementation Specifics

Software Product/Option: *AIX Performance Aide/ Local Performance Analysis & Control Commands*.

Standards Compliance: None.

Files

/usr/bin/fdpr	Contains the **fdpr** command.
program	Name of the unstripped executable to be optimized.
*program***.save**	Saved version of the original executable program.
*program***.nprof**	Name of the profile file.
*program***.instr**	Name of the instrumented version of program.
*program***.fdpr**	Default name of optimized executable output file.
*program***.instr.dis_text**	Default disassembly file in ASCII format produced by **-disasm** flag after instrumentation phase.
*program***.fdpr.dis_text**	Default disassembly file in ASCII format produced by **-disasm** flag after optimization phase.
*program***.instr.dis_data**	Default disassembly file in ASCII format produced by **-disasm_data** flag after instrumentation phase.
*program***.fdpr.dis_data**	Default disassembly file in ASCII format produced by **-disasm_data** flag after optimization phase.
*program***.instr.dis_bss**	Default disassembly file in ASCII format produced by **-disasm_bss** flag after instrumentation phase.
*program***.fdpr.dis_bss**	Default disassembly file in ASCII format produced by **-disasm_bss** flag after optimization phase.
*program***.instr.mapper**	Default mapping file in ASCII format produced by **-map** flag after instrumentation phase.
*program***.fdpr.mapper**	Default mapping file in ASCII format produced by **-map** flag after optimization phase.
*program***.ncounts**	Default profile counters file in ASCII format produced by **-profcount** flag.

Related Information

The **dbx** command.

Restructuring executable programs with the fdpr program in *Performance management*.

The xlC compiler.

---

fencevsd Command

Purpose

Prevents an application running on a node or group of nodes from accessing a virtual shared disk or group of virtual shared disks.

Syntax

fencevsd {**–a** | **–v** *vsd_name_list*} **–n** *node_list*

Description

Under some circumstances, the system may believe a node has stopped functioning and begin recovery procedures, when the node is actually operational, but cut off from communication with other nodes running the same application. In this case, the problem node must not be allowed to serve requests for the virtual shared disks it normally serves until recovery is complete and the other nodes running the application recognize the problem node as operational. The **fencevsd** command prevents the problem node from filling requests for its virtual shared disks.

This command can be run from any node in the RSCT peer domain where the recoverable virtual shared disk subsystem is running.

Flags

–a Specifies all virtual shared disks.

–v *vsd_name_list*
Specifies one or more virtual shared disk names, separated by commas.

–n *node_list* Specifies one or more node numbers, separated by commas.

Parameters

logical_volume_name
Is the name of the logical volume you want to specify as a 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 a 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 virtual shared disk. This name must be unique within the RSCT peer domain, and, in order to avoid possible future naming conflicts, should also be unique across the overall cluster. 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 specify a *vsd_name* that is already the name of another device, the **cfgvsd** command will be unsuccessful for that virtual shared disk. This error 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).

Security

You must have **root** authority to run this command.

Restrictions

You must issue this command from a node in the peer domain that has an active recoverable virtual shared disk subsystem.

Examples

To fence the virtual shared disks vsd1 and vsd2 from node 5, enter:

```
fencevsd -v vsd1,vsd2 -n 5
```

Location

/opt/rsct/vsd/bin/fencevsd

Related Information

Commands: **lsvsd**, **unfencevsd**

Books: *RSCT for AIX 5L: Managing Shared Disks* for more information on the recoverable virtual shared disk subsystem and how you can use the **fencevsd** and **unfencevsd** commands to preserve data integrity during application recovery.

---

ff Command

Purpose

Lists the file names and statistics for a file system.

Syntax

ff [**-a** *Number*] [**-c** *Number*] [**-l**] [**-l**] [**-m** *Number*] [**-n** *File*] [**-o** *Options*] [**-p** *Prefix*] [**-s**] [**-u**] [**-V** *VFSName*] [**-i** *I-Number* [**,***I-Number ...*]] [*FileSystem* | *DeviceName*]

Description

The **ff** command reads the i-nodes in the file system specified by the *FileSystem* parameter and then writes information about them to standard output. It assumes the *FileSystem* is a file system, which is referenced in the **/etc/filesystems** file, and saves i-node data for files specified by flags.

The output from the **ff** command consists of the path name for each requested i-node number, in addition to other file information that you can request using the flags. The output is listed in order by i-node number, with tabs between all fields. The default line produced by the **ff** command includes the path name and i-node number fields. With all flags enabled, the output fields include path name, i-node number, size, and UID (user ID).

The *Number* parameter is a decimal number that specifies a number of days. It is prefixed by a + or - (plus or minus sign). Therefore, +3 means more than 3 days, -3 means less than 3 days, and 3 means 3 days, where a day is defined as a 24-hour period.

The **ff** command lists only a single path name out of many possible ones for an i-node with more than one link, unless you specify the **-l** flag. With the **-l** flag, the **ff** command lists all links.

Flags

-a *Number*	Displays the file if it has been accessed within the number of days specified by the *Number* parameter.
-c *Number*	Displays the file if its i-node has been changed within the number of days specified by the *Number* parameter.
-i *I-Number*	Displays the files corresponding to the i-node numbers specified by the *I-Number* parameter. The i-node numbers listed must be separated by a comma.
-l	(This flag is an uppercase i.) Does not display the i-node after each path name.
-l	(This flag is a lowercase L.) Additionally displays a list of pathnames for files with more than one link.
-m *Number*	Displays the file if it has been modified within the number of days specified by the *Number* parameter.

-n *File* Displays the file if it has been modified more recently than the file specified by the *File* parameter.

-o *Options* Specifies a comma-separated list of implementation-specific options for a virtual file system.

The following options are specific to the enhanced journaled file system (JFS2):

-o snapshot=*snapName* Specifies the name of the internal snapshot subject to the **ff** command. The file system owning the snapshot must be mounted.

-p *Prefix* Adds the prefix specified by the *Prefix* parameter to each path name. The default prefix is **.** (dot).

-s Writes the file size, in bytes, after each path name.

-u Writes the owner's login name after each path name.

-V *VFSName* Instructs the **ff** command to assume the file system is of type *VFSName*, overriding the value in the **/etc/filesystems** file.

Security

Attention RBAC users and Trusted AIX users: This command can perform privileged operations. Only privileged users can run privileged operations. For more information about authorizations and privileges, see Privileged Command Database in *Security*. For a list of privileges and the authorizations associated with this command, see the **/etc/security/privcmds** database file.

Examples

1. To list the path names of all files in a given file system, enter:

   ```
   ff    -I    /dev/hd0
   ```

   This displays the path names of the files on the `/dev/hd0` device. If you do not specify the **-I** flag, the **ff** command also displays the i-node number of each file.
2. To list files that have been modified recently, enter:

   ```
   ff    -m    -2    -u    /dev/hd0
   ```

   This displays the path name, i-node number, and owner's user name (the `-u` flag) of each file on the `/dev/hd0` device that has been modified within the last two days (`-m -2`).
3. To list files that have *not* been used recently, enter:

   ```
   ff    -a    +30    /dev/hd0
   ```

   This displays the path name and i-node of each file that was last accessed more than 30 days ago (`-a +30`).
4. To find out the paths corresponding to certain i-node numbers, enter:

   ```
   ff    -l    -i    451,76    /dev/hd0
   ```

   This displays all the path names (`-l`) associated with i-nodes `451` and `76`.

Files

/etc/vfs Contains descriptions of virtual file system types.

/etc/filesystems Lists the known file systems and defines their characteristics.

Related Information

The **find** command, **ncheck** command.

The File systems in*Operating system and device management* explains file system types, management, structure, and maintenance.

---

fg Command

Purpose

Runs jobs in the foreground.

Syntax

fg [*JobID*]

Description

If job control is enabled (see ″Job control in the Korn shell or POSIX shell″ in *Operating system and device management*), the **fg** command moves a background job in the current environment into the foreground. Use the *JobID* parameter to indicate a specific job to be run in the foreground. If this parameter is not supplied, the **fg** command uses the job most recently suspended, placed in the background, or run as a background job.

The *JobID* parameter can be a process ID number, or you can use one of the following symbol combinations:

%*Number*	Refers to a job by the job number.
%*String*	Refers to a job whose name begins with the specified string.
%?*String*	Refers to a job whose name contains the specified string.
%+ OR **%%**	Refers to the current job.
%-	Refers to the previous job.

Using the **fg** command to place a job into the foreground removes the job's process ID from the list of those known by the current shell environment.

The **/usr/bin/fg** command does not work when operating in its own command execution environment, because that environment does not have applicable jobs to manipulate. For this reason, the **fg** command is implemented as a Korn shell or POSIX shell regular built-in command.

Exit Status

The following exit values are returned:

0	Successful completion.
>0	An error occurred.

If job control is disabled, the **fg** command exits with an error, and no job is placed in the foreground.

Examples

If the output of the **jobs -l** command shows the following job running in the background:

```
[1] + 16477RunningSleep 100 &
```

use the process ID to run the `sleep 100 &` command in the foreground by entering:

```
fg 16477
```

The screen displays:

```
sleep
```

Files

/usr/bin/ksh	Contains the Korn shell **fg** built-in command.
/usr/bin/fg	Contains the **fg** command.

Related Information

The **bg** command, **csh** command, **jobs** command, **kill** command, **wait** command.

Job control in the Korn shell or POSIX shell in *Operating system and device management*.

---

fgrep Command

Purpose

Searches a file for a literal string.

Syntax

fgrep [**-h**] [**-i**] [**-s**] [**-v**] [**-w**] [**-x**] [**-y**] [[**-b**] [**-n**] | [**-c** | **-l** | **-q**]] [**-p***Separator*] {*Pattern* | **-e***Pattern* | **-f***StringFile*} [*File...*]

Description

The **fgrep** command searches the input files specified by the *File* Parameter (standard input by default) for lines matching a pattern. The **fgrep** command searches specifically for *Pattern* parameters that are fixed strings. The **fgrep** command displays the file containing the matched line if you specify more than one file in the *File* parameter.

The **fgrep** command differs from the **grep** and **egrep** commands because it searches for a string instead of searching for a pattern that matches an expression. The **fgrep** command uses a fast and compact algorithm. The $, *, [, |, (,) and \ characters are interpreted literally by the **fgrep** command. These characters are not interpreted as parts of a regular expression, as they would be in the **grep** and **egrep** command. Since these characters have special meaning to the shell, the entire string should be enclosed in single quotes ('...'). If no files are specified, the **fgrep** command assumes standard input. Normally, each line found is copied to the standard output. The filename is printed before each line found if there is more than one input file.

Notes:

1. The **fgrep** command is the same as the **grep** command with the **-F** flag, except that error and usage messages are different and the **-s** flag functions differently.
2. Lines are limited to 2048 bytes.
3. Paragraphs (under the **-p** flag) are currently limited to a length of 5000 characters.
4. Do not run the **grep** command on a special file because it produces unpredictable results.
5. Input lines should not contain the NULL character.
6. Input files should end with the new line character.
7. Although some flags can be specified simultaneously, some flags override others. For example, if you specify **-l** and **-n** together, only file names are written to standard output.

Flags

-b	Precedes each line by the block number on which it was found. Use this flag to help find disk block numbers by context. The **-b** flag cannot be used with input from stdin or pipes.
-c	Displays only a count of matching lines.
-e *Pattern*	Specifies a pattern. This works like a simple pattern but is useful when the pattern begins with a **-** (minus sign).
-f *StringFile*	Specifies a file that contains strings.
-h	Suppresses file names when multiple files are being processed.
-i	Ignores the case of letters when making comparisons.
-l	Lists just the names of files (once) with matching lines. Each file name is separated by a new line character.
-n	Precedes each line with its relative line number in the file.
-p*Separator*	Displays the entire paragraph containing matched lines. Paragraphs are delimited by paragraph separators, as specified by the *Separator* parameter, which are patterns in the same form as the search pattern. Lines containing the paragraph separators are used only as separators; they are never included in the output. The default paragraph separator is a blank line.
-q	Suppresses all writing to standard output, regardless of matching lines. Exits with a 0 status if an input line is selected.
-s	Displays only error messages. This is useful for checking status.
-v	Displays all lines except those that match the specified pattern.
-w	Does a word search.
-x	Displays lines that match the pattern exactly with no additional characters.
-y	Ignores the case of letters when making comparisons.

Exit Status

This command returns the following exit values:

0	A match was found.
1	No match was found.
>1	A syntax error was found or a file was inaccessible (even if matches were found).

Examples

1. To search several files for a simple string of characters:

   ```
   fgrep strcpy *.c
   ```

   This searches for the string `strcpy` in all files in the current directory with names ending in the `.c` character string.

2. To count the number of lines that match a pattern:

   ```
   fgrep -c "{" pgm.c
   fgrep -c "}" pgm.c
   ```

   This displays the number of lines in `pgm.c` that contain left and right braces.

   If you do not put more than one { (left brace) or one } (right brace) on a line in your C programs, and if the braces are properly balanced, the two numbers displayed are usually the same if the proper conditions are met. If the numbers are not the same, you can display the lines that contain braces in the order that they occur in the file with:

   ```
   egrep {\|} pgm.c
   ```

3. To display the names of files that contain a pattern:

   ```
   fgrep -l strcpy *.c
   ```

This searches the files in the current directory that end with `.c`and displays the names of those files that contain the `strcpy` string.

Files

/usr/bin/fgrep	Contains the **fgrep** command.
/bin/fgrep	Symbolic link to the **fgrep** command.

Related Information

The **ed** command, **egrep** command, **grep** command, **sed** command.

Files in *Operating system and device management* introduces you to files and the way you can work with them.

Input and output redirection in *Operating system and device management* describes how the operating system processes input and output.

---

file Command

Purpose

Determines the file type.

Syntax

To Classify the File Type

file [**-m** *MagicFile*] [**-d**] [**-h**] [**-i**] [**-M** *MagicFile*] [**-f** *FileList*] [*File...*]

To Check the Magic File for Format Errors

file -c [**-m** *MagicFile*]

Description

The **file** command reads the files specified by the *File* parameter or the *FileList* variable, performs a series of tests on each file, and attempts to classify them by type. The command then writes the file types to standard output. The file can be regular file, directory, FIFO(named pipe), block special, character special, symbolic link or sockets type.

- If it is a regular file and of zero length, it is identified as an empty file.
- If the file is a symbolic link, by default, the link is followed by file the symbolic link refers to.

If a file appears to be in ASCII format, the **file** command examines the first 1024 bytes and determines the file type. If a file does not appear to be in ASCII format, the **file** command further attempts to distinguish a binary data file from a text file that contains extended characters.

If the *File* parameter specifies an executable or object module file and the version number is greater than 0, the **file** command displays the version stamp. The **ld** command explains the use of **a.out** files.

The **file** command uses the **/etc/magic** file to identify files that have some sort of a magic number; that is, any file containing a numeric or string constant that indicates type.

If the file does not exist, cannot be read or its file status could not be determined then, it is not considered as an error that affects the exit status. The output indicates that the file was processed but the type could not be determined.

When the **-i** flag is used, the following format shall be used to identify each operand, *file* specified:
`"%s: %s\n", file, type`

The values for *type* are unspecified except that in the POSIX locale, if *file* is identified as one of the types listed in the following table, *type* shall contain (but is not limited to) the corresponding string. Each space shown in the strings shall be exactly one *space*.

Table 5. File Utility Output Strings

If *file* **is a:**	*type* **shall contain the string:**
Directoy	directory
FIFO	fifo
Socket	socket
Block special	block special
Character special	character special
Executable binary	executable
Empty regular file	empty
Symbolic link	symbolic link to
ar archive library	archive
Extended *cpio* format	cpio archive
Extended *tar* format	tar archive
Shell script	commands text
C-language source	c program text
FORTRAN source	fortran program text

If *file* is identified as a symbolic link, the following alternative output format shall be used:
`"%s: %s %s\n", file, type, contents of link"`

If the file named by the *file* operand does not exist or cannot be read, the string `cannot open` shall be included as part of the *type* field, but this shall not be considered an error that affects the exit status. If the type of the file named by the *file* operand cannot be determined, the string `data` shall be included as part of the *type* field, but this shall not be considered an error that affects the exit status.

Flags

-c Checks the specified magic file (the **/etc/magic** file, by default) for format errors. This validation is not normally done. File typing is not done under this flag.

-d Applies any default system tests to the file.

-f *FileList* Reads the specified file list. The file must list one file per line and must not contain leading or trailing spaces.

-h When a symbolic link is encountered, identifies the file as a symbolic link. If the **-h** flag is not specified and *file* is a symbolic link that refers to a nonexistent file, *file* shall identify the file as a symbolic link, as if the **-h** flag had been specified.

-i If a file is a regular file, does not attempt to classify the type of the file further, but identifies the file as specified in “Description” on page 486.

-m *MagicFile* Specifies the file name of the magic file (the **/etc/magic** file, by default).

-M *MagicFile* Specifies the name of a file containing tests that shall be applied to a file in order to classify it. No default system tests shall be applied.

Exit Status

This command returns the following exit values:

0 Successful completion.
>0 An error occurred.

Examples

1. To display the type of information a file contains, enter:

   ```
   file myfile
   ```

   This displays the file type of *myfile* (such as directory, data, ASCII text, C-program source, and archive).
2. To display the type of each file named in a list of file names, enter:

   ```
   file -f filenames
   ```

   This displays the type of each file named in the `filenames` list. Each file name must appear alone on a line.

Note: To get customized messages from the **file** command, use a separate magic file with the **-m** option. It is not advisable to edit the read-only **/etc/magic** file.

Files

/usr/bin/file	Contains the **file** command.
/etc/magic	Contains the file type database.

Related Information

The **find** command, **ld** command.

Files in *Operating system and device management* describes files, file types, and how to name files.

Input and Output Redirection Overview in *Operating system and device management* describes how the operating system processes input and output.

File and directory access modes in *Operating system and device management* introduces file ownership and permissions to access files and directories.

---

filemon Command

Purpose

Monitors the performance of the file system, and reports the I/O activity on behalf of logical files, virtual memory segments, logical volumes, and physical volumes.

Syntax

filemon [**-d**] [**-i** *Trace_File* **-n** *Gensyms_File*] [**-o** *File*] [**-O** *Levels*] [**-P**] [**-T** *n*] [**-u**] [**-v**] [**-@** [*WparList* | **ALL**]

Description

The **filemon** command monitors a trace of file system and I/O system events, and reports on the file and I/O access performance during that period.

In its normal mode, the **filemon** command runs in the background while one or more application programs or system commands are being executed and monitored. The **filemon** command automatically starts and monitors a trace of the program's file system and I/O events in real time. By default, the trace is started immediately; optionally, tracing may be deferred until the user issues a **trcon** command. The user can issue **trcoff** and **trcon** commands while the **filemon** command is running in order to turn off and on monitoring, as desired. When tracing is stopped by a **trcstop** command, the **filemon** command generates an I/O activity report and exits.

The **filemon** command can also process a trace file that has been previously recorded by the trace facility. The file and I/O activity report will be based on the events recorded in that file.

To provide a more complete understanding of file system performance for an application, the **filemon** command monitors file and I/O activity at four levels:

Logical file system	The **filemon** command monitors logical I/O operations on logical files. The monitored operations include all **read**, **write**, **open**, and **lseek** system calls, which may or may not result in actual physical I/O, depending on whether or not the files are already buffered in memory. I/O statistics are kept on a per-file basis. Calls to Asynchronous I/O system calls are not monitored by the **filemon** command, so the **filemon** logical file report does not include asynchronous I/O (AIO) requests.
Virtual memory system	The **filemon** command monitors physical I/O operations (that is, paging) between segments and their images on disk. I/O statistics are kept on a per-segment basis.
Logical volumes	The **filemon** command monitors I/O operations on logical volumes. I/O statistics are kept on a per-logical-volume basis.
Physical volumes	The **filemon** command monitors I/O operations on physical volumes. At this level, physical resource utilizations are obtained. I/O statistics are kept on a per-physical-volume basis.

Any combination of the four levels can be monitored, as specified by the command line flags. By default, the **filemon** command only monitors I/O operations at the virtual memory, logical volume, and physical volume levels. These levels are all concerned with requests for real disk I/O.

The **filemon** command writes its report to standard output or to a specified file. The report begins with a summary of the I/O activity for each of the levels being monitored and ends with detailed I/O activity statistics for each of the levels being monitored. Summary and detailed report contents are described in the Reports section.

Notes:

1. The reports produced by the **filemon** command can be quite long. Consequently, the **-o** option should usually be used to write the report to an output file. When a physical device is opened and accessed directly by an application, only reads and writes of complete 512-byte blocks are reflected in the report. "Short" reads and writes, used by the device driver to issue device commands and read device status, are ignored. CD-ROMs do not have concentric "tracks" or "cylinders," as in hard files. (There is one spiral track.) Consequently, it is not possible to report seek distance statistics for CD-ROMs in terms of cylinders.
2. The **-u** flag is used to generate reports on files opened prior to the start of the **trace** daemon. Some of this data can be useful, but much of it applies to daemons and other unrelated activity. This background information can be overwhelming, especially on large systems. If the **/unix** file and the running kernel are not the same, then the kernel addresses will be incorrect, causing the **filemon**

command to exit. When using the **filemon** command from within a shell script, allow for a slight delay prior to viewing the contents of the **filemon** output file. The **filemon** command may take a few seconds to produce this report.

System Trace Facility

The **filemon** command obtains raw I/O performance data using the system trace facility. Currently, the trace facility only supports one output stream. Consequently, only one **filemon** or trace process can be active at a time. If another **filemon** or trace process is already running, the **filemon** command responds with the message:

```
/dev/systrace: Device busy
```

While monitoring very I/O-intensive applications, the **filemon** command may not be able to consume trace events as fast as they are produced in real time. When that happens, the error message:

```
Trace kernel buffers overflowed, N missed entries
```

will be displayed on `stderr`, indicating how many trace events were lost while the trace buffers were full. The **filemon** command will continue monitoring I/O activity, but the accuracy of the report will be diminished to some unknown degree. One way to prevent overflow is to monitor fewer levels of the file and I/O subsystems: the number of trace events generated is proportional to the number of levels monitored. Additionally, the trace buffer size can be increased using the **-T** option, to accommodate larger bursts of trace events before overflow. Remember that increasing the trace buffer size will result in more pinned memory, and therefore may effect I/O and paging behavior.

In memory-constrained environments (where demand for memory exceeds supply), the **-P** option can be used to pin the text and data pages of the real-time **filemon** process in memory so the pages cannot be swapped out. If the **-P** option is not used, allowing the **filemon** process to be swapped out, the progress of the **filemon** command may be delayed to the point where it cannot process trace events fast enough. This situation leads to trace buffer overflow as described above. Of course, pinning this process takes memory away from the application (although the **filemon** command is not a large program, its process image can consume up to 500KB).

The **-i** *Trace_File* and **-n** *Gensyms_File* flags allow for offline processing by **filemon** of trace data files created by the **trace** command. Both flags must be supplied if either is present. These flags are useful when it is necessary to postprocess a trace file from a remote machine or perform the trace data collection at one time and postprocess it at another time. The flags are also useful when system load is high and trace hooks are being missed by **filemon**.

The **gensyms** file (containing filesystem information) must be used from the machine that the trace came from. Also, it is wise to run **gensyms** at close to the same time that the system trace file is created, so that the system configuration is the same for both.

Trace hooks relevant to **filemon** must be collected by the **trace** command and are specified by the **trace -j** flag. The relevant trace hooks are listed when **filemon** is invoked with the **-v** flag. The **gensyms** command with **-f**option is then executed, with its output saved in *Gensyms_File* to collect additional information for **filemon**. The **-f** option is used with the gensyms command to collect the device information for physical and logical volumes. It is also used to get the virtual file system information used by offline **filemon**. Then this file and the *Gensyms_File* may be provided to **filemon**.

Reports

Each report generated by the **filemon** command has a header that identifies the date, the machine ID, and the length of the monitoring period, in seconds. The CPU utilization during the monitoring period is also reported.

Next, summary reports are generated for each of the file system levels being monitored. By default, the logical file and virtual memory reports are limited to the 20 most active files and segments, respectively, as

measured by the total amount of data transferred. If the **-v** flag has been specified, activity for all files and segments is reported. There is one row for each reported file, segment, or volume. The columns in each row for the four summary reports are described in the following lists:

Most Active Files Report

Column	Description
`#MBS`	Total number of megabytes transferred to/from file. The rows are sorted by this field, in decreasing order.
`#opns`	Number of times the file was opened during measurement period.
`#rds`	Number of read system calls made against file.
`#wrs`	Number of write system calls made against file.
`file`	Name of file (full path name is in detailed report).
`volume:inode`	Name of volume that contains the file, and the file's i-node number. This field can be used to associate a file with its corresponding persistent segment, shown in the virtual memory I/O reports. This field may be blank; for example, for temporary files created and deleted during execution.

Most Active Segments Report

Column	Description
`#MBS`	Total number of megabytes transferred to/from segment. The rows are sorted by this field, in decreasing order.
`#rpgs`	Number of 4096-byte pages read into segment from disk (that is, page).
`#wpgs`	Number of 4096-byte pages written from segment to disk (page out).
`segid`	Internal ID of segment.
`segtype`	Type of segment: working segment, persistent segment (local file), client segment (remote file), page table segment, system segment, or special persistent segments containing file system data (log, root directory, `.inode`, `.inodemap`, `.inodex`, `.inodexmap`, `.indirect`, `.diskmap`).
`volume:inode`	For persistent segments, name of volume that contains the associated file, and the file's inode number. This field can be used to associate a persistent segment with its corresponding file, shown in the file I/O reports. This field is blank for non-persistent segments. **Note:** The virtual memory analysis tool, **svmon** can be used to display more information about a segment, given its segment ID (segid), as follows: `svmon -S <segid>`

Most Active Logical Volumes Report

Column	Description
`util`	Utilization of the volume (fraction of time busy). The rows are sorted by this field, in decreasing order.
`#rblk`	Number of 512-byte blocks read from the volume.
`#wblk`	Number of 512-byte blocks written to the volume.
`KB/sec`	Total transfer throughput, in Kilobytes per second.
`volume`	Name of volume.
`description`	Contents of volume: either a file system name, or logical volume type (paging, jfslog, boot, or sysdump). Also, indicates if the file system is fragmented or compressed.

Most Active Physical Volumes Report

Column	Description
`util`	Utilization of the volume (fraction of time busy). The rows are sorted by this field, in decreasing order.
`#rblk`	Number of 512-byte blocks read from the volume.

#wblk	Number of 512-byte blocks written to the volume.
KB/sec	Total volume throughput, in Kilobytes per second.
volume	Name of volume.
description	Type of volume, for example, 120MB disk, 355MB SCSI, or CDROM SCSI. **Note:** Logical volume I/O requests start before, and end after, physical volume I/O requests. For that reason, total logical volume utilization will appear to be higher than total physical volume utilization.

Finally, detailed reports are generated for each of the file system levels being monitored. By default, the logical file and virtual memory reports are limited to the 20 most active files and segments, respectively, as measured by the total amount of data transferred. If the **-v** flag is specified, activity for all files and segments is reported. There is one entry for each reported file, segment, or volume.

Some of the fields report a single value, others report statistics that characterize a distribution of many values. For example, response time statistics are kept for all read or write requests that were monitored. The average, minimum, and maximum response times are reported, as well as the standard deviation of the response times. The standard deviation is used to show how much the individual response times deviated from the average. Roughly two-thirds of the sampled response times are between average - standard deviation and average + standard deviation. If the distribution of response times is scattered over a large range, the standard deviation will be large compared to the average response time. The four detailed reports are described in the following lists:

Detailed File Stats Report

Column	Description
FILE	Name of the file. The full path name is given, if possible.
volume	Name of the logical volume/file system containing the file.
inode	I-node number for the file within its file system.
opens	Number of times the file was opened while monitored.
total bytes xfrd	Total number of bytes read/written to/from the file.
reads	Number of read calls against the file.
read sizes (bytes)	The read transfer-size statistics (avg/min/max/sdev), in bytes.
read times (msec)	The read response-time statistics (avg/min/max/sdev), in milliseconds.
writes	Number of write calls against the file.
write sizes (bytes)	The write transfer-size statistics.
write times (msec)	The write response-time statistics.
seeks	Number of **lseek** subroutine calls.

Detailed VM Segment Stats Report

Column	Description
SEGMENT	Internal segment ID.
segtype	Type of segment contents.
segment flags	Various segment attributes.
volume	For persistent segments, the name of the logical volume containing the corresponding file.
inode	For persistent segments, the i-node number for the corresponding file.
reads	Number of 4096-byte pages read into the segment (that is, paged in).
read times (msec)	The read response-time statistics (avg/min/max/sdev), in milliseconds.
read sequences	Number of read sequences. A sequence is a string of pages that are read (paged in) consecutively. The number of read sequences is an indicator of the amount of sequential access.
read seq. lengths	Statistics describing the lengths of the read sequences, in pages.
writes	Number of pages written from the segment (that is, paged out).
write times (msec)	Write response time statistics.

write sequences	Number of write sequences. A sequence is a string of pages that are written (paged out) consecutively.
write seq.lengths	Statistics describing the lengths of the write sequences, in pages.

Detailed Logical/Physical Volume Stats Reports

Column	Description
VOLUME	Name of the volume.
description	Description of the volume. (Describes contents, if discussing a logical volume; describes type, if dealing with a physical volume.)
reads	Number of read requests made against the volume.
read sizes (blks)	The read transfer-size statistics (avg/min/max/sdev), in units of 512-byte blocks.
read times (msec)	The read response-time statistics (avg/min/max/sdev), in milliseconds.
read sequences	Number of read sequences. A sequence is a string of 512-byte blocks that are read consecutively and indicate the amount of sequential access.
read seq. lengths	Statistics describing the lengths of the read sequences, in blocks.
writes	Number of write requests made against the volume.
write sizes (blks)	The write transfer-size statistics.
write times (msec)	The write-response time statistics.
write sequences	Number of write sequences. A sequence is a string of 512-byte blocks that are written consecutively.
write seq. lengths	Statistics describing the lengths of the write sequences, in blocks.
seeks	Number of seeks that preceded a read or write request; also expressed as a percentage of the total reads and writes that required seeks.
seek dist (blks)	Seek distance statistics, in units of 512-byte blocks. In addition to the usual statistics (avg/min/max/sdev), the distance of the initial seek operation (assuming block 0 was the starting position) is reported separately. This seek distance is sometimes very large, so it is reported separately to avoid skewing the other statistics.
seek dist (cyls)	(Hard files only.) Seek distance statistics, in units of disk cylinders.
time to next req	Statistics (avg/min/max/sdev) describing the length of time, in milliseconds, between consecutive read or write requests to the volume. This column indicates the rate at which the volume is being accessed.
throughput	Total volume throughput, in Kilobytes per second.
utilization	Fraction of time the volume was busy. The entries in this report are sorted by this field, in decreasing order.

Flags

-i *Trace_File*
Reads the I/O trace data from the specified *Trace_File*, instead of from the real-time trace process. The **filemon** report summarizes the I/O activity for the system and period represented by the trace file.

For the report to be accurate, the trace file must contain all the hooks required by the **filemon** command.

The **-n** option must also be specified.

-n *Gensyms_File*
Specifies a *Gensyms_File* for offline trace processing. This file is created by running the gensyms command with **-f** option and redirecting the output to a file, as follows:

```
gensyms -f > file
```

The **-i** option must also be specified.

-o *File*
Writes the I/O activity report to the specified *File*, instead of to the **stdout** file.

-d
Starts the **filemon** command, but defers tracing until the **trcon** command has been executed by the user. By default, tracing is started immediately.

-T *n* Sets the kernel's trace buffer size to *n* bytes. The default size is 64 000 bytes per CPU. The buffer size can be increased to accommodate larger bursts of events, if any. (A typical event record size is 30 bytes.)
Note: The trace driver in the kernel uses double buffering, so in fact there will be two buffers allocated of size *n* bytes. Also, note that these buffers are pinned in memory, so they are not subject to paging. Large buffers may affect the performance of paging and other I/O.

-P Pins monitor process in memory. The **-P** flag causes the **filemon** command's text and data pages to be pinned in memory for the duration of the monitoring period. This flag can be used to ensure that the real-time **filemon** process is not paged out when running in a memory-constrained environment.

-v Prints extra information in the report. The most significant effect of the **-v** flag is that all logical files and all segments that were accessed are included in the I/O activity report, instead of only the 20 most active files and segments.

-O *Levels* Monitors only the specified file system levels. Valid level identifiers are:

lf Logical file level.

vm Virtual memory level. This report is not available inside a workload partition (WPAR) in online mode or in the global WPAR with the **-@** *WparList* option.

lv Logical volume level. This report is not available inside a WPAR in online mode or in the global WPAR with the **-@** *WparList* option.

pv Physical volume level. This report is not available inside a WPAR in online mode or in the global WPAR with the **-@** *WparList* option.

all Short for **lf, vm, lv, pv**. Short for **lf** inside a WPAR in online mode or in the global WPAR with the **-@** *WparList* option.

The **vm**, **lv**, and **pv** levels are implied by default when you run the **filemon -O** command in the global WPAR without the **-@** flag. The **lf** level is implied by default when you run the **filemon -O** command in a WPAR or when you use the **-@** flag.

-u Reports on files that were opened prior to the start of the **trace** daemon. The process ID (PID) and the file descriptor (FD) are substituted for the file name.
Note: Since PIDs and FDs are reusable, it is possible to see different files reported with the same name field.

-@ [*WparList* | **ALL**] Reports are limited to the list of WPARs passed by the argument.

Examples

1. To monitor the physical I/O activity of the virtual memory, logical volume, and physical volume levels of the file system, enter:

   ```
   filemon
   ```

   The **filemon** command automatically starts the system trace and puts itself in the background. After this command, enter the application programs and system commands to be run at this time, then enter:

   ```
   trcstop
   ```

   After the **trcstop** command is issued, the I/O activity report is displayed on standard output (but will probably scroll off the screen). The virtual memory I/O report will be limited to the 20 segments that incurred the most I/O.

2. To monitor the activity at all file system levels, and write the report to the `fmon.out` file, enter:

   ```
   filemon -o fmon.out -O all
   ```

   The **filemon** command automatically starts the system trace and puts itself in the background. After this command, enter the application programs and system commands to be run at this time, then enter:

```
trcstop
```

After the **trcstop** command is issued, the I/O activity report is written to the `fmon.out file`. All four levels of the file and I/O system (the logical file, virtual memory, logical volume, and physical volume levels) will be monitored. The logical file and virtual memory I/O reports will be limited to the 20 files and segments (respectively) that incurred the most I/O.

3. To monitor the activity at all file system levels and write a verbose report to the `fmon.out` file, enter:

```
filemon -v -o fmon.out -O all
```

The **filemon** command automatically starts the system trace and puts itself in the background. After this command, enter the application programs and system commands to be run at this time, then enter:

```
trcstop
```

This example is similar to the previous example, except a verbose report is generated on the `fmon.out` file. The primary difference is that the **filemon** command will indicate the steps it is taking to start up the trace, and the summary and detailed reports will include all files and segments that incurred any I/O (there may be many), instead of just the top 20.

4. To report on I/O activity captured by a previously recorded trace session, enter:

```
filemon -i trcfile | pg
```

In this example, the **filemon** command reads file system trace events from the input file `trcfile`. Since the trace data is already captured on a file, the **filemon** command does not put itself in the background to allow application programs to be run. After the entire file is read, an I/O activity report for the virtual memory, logical volume, and physical volume levels will be displayed on standard output (which, in this example, is piped to `pg`).

5. To monitor the I/O activity for logical and physical volumes only, while controlling the monitored intervals using the **trcon** and **trcoff** commands, enter:

```
filemon -d -o fmon.out -O pv,lv
```

The **filemon** command automatically starts the system trace and puts itself in the background. After this command, you can enter the unmonitored application programs and system commands to be run at this time, then enter:

```
trcon
```

After this command, you can enter the monitored application programs and system commands to be run at this time, then enter:

```
trcoff
```

After this command, you can enter the unmonitored application programs and system commands to be run at this time, then enter:

```
trcon
```

After this command, you can enter the monitored application programs and system commands to be run at this time, then enter:

```
trcstop
```

In this example, the **-O** flag is used to restrict monitoring to logical and physical volumes only. Only those trace events that are relevant to logical and physical volumes are enabled. Also, as a result of using the **-d** flag, monitoring is initially deferred until the **trcon** command is issued. System tracing can be intermittently disabled and reenabled using the **trcoff** and **trcon** commands, so that only specific intervals are monitored.

6. To run **filemon** in offline mode, run the **trace** and **gensyms** commands separately, then use the output from those commands as input to the **filemon** command, as follows:

```
trace -a -T 768000 -L 10000000 -o trace.out -j 000,000,001,002,003,005,006,139,102,10C,106,00A,107,
101,104,10D,15B,12E,130,163,19C,154,3D3,1BA,1BE,1BC,10B,221,1C9,222,228,232,45B
```

Run the monitored application programs and system commands, then enter:

```
trcstop
```

Create the **gensyms** file:

```
gensyms -f > gensyms.out
```

Then run **filemon** with both **-i** and **-n** flags:

```
filemon -i trace.out -n gensyms.out -O all
```

Related Information

The **svmon** command, **trcrpt** command, **trcstop** command.

The **lseek** subroutine.

Monitoring disk I/O in *Performance management*.

---

fileplace Command

Purpose

Displays the placement of file blocks within logical or physical volumes.

Syntax

fileplace [{ **-l** | **-p** } [**-i**] [**-v**]] *File* | [**-m** *LogicalVolumeName*]

Description

The **fileplace** command displays the placement of a specified file within the logical or physical volumes containing the file.

By default, the **fileplace** command lists to standard output the ranges of logical volume fragments allocated to the specified file. The order in which the logical volume fragments are listed corresponds directly to their order in the file. A short header indicates the file size (in bytes), the name of the logical volume in which the file lies, the block size (in bytes) for that volume, the fragment size in bytes, and the compression, indicating if the file system is compressed or not.

Occasionally, portions of a file may not be mapped to any fragments in the volume. These areas, whose size is an integral number of fragments, are implicitly zero-filled by the file system. The **fileplace** command indicates which areas in a file have no allocated fragments.

Optionally, the **fileplace** command also displays:

- Statistics indicating the degree to which the file is spread within the volume.
- The indirect block addresses for the file.
- The file's placement on physical (as opposed to logical) volume, for each of the physical copies of the file.

Notes:

1. The **fileplace** command is not able to display the placement of remote Network File System (NFS) files. If a remote file is specified, the **fileplace** command returns an error message. However, the placement of the remote file can be displayed if the **fileplace** command is run directly on the file server.

2. The **fileplace** command reads the file's list of blocks directly from the logical volume on disk. If the file is newly created, extended, or truncated, the file system information may not yet be on the disk when the **fileplace** command is run. Use the **sync** command to flush the file information to the logical volume.
3. There is no Indirect/Double Indirect blocks concept in JFS2 filesystem. The file is represented in terms of extents. Therefore the size of the maximum extent depends on the aggregate block size. With a 512 byte aggregate block size (the smallest allowable), the maximum extent is 512*(2^ 24-1) bytes long (slightly under 8G). With a 4096 byte aggregate block size (the largest allowable), the maximum extent is 4096*(2^ 24-1) bytes long (slightly under 64G).

   These limits apply only to a single extent; in no way do they have any limiting effects on overall file sizes.

Flags

-i Displays the indirect blocks for the file, if any. The indirect blocks are displayed in terms of either their logical or physical volume block addresses, depending on whether the **-l** or **-p** flag is specified.

-l Displays file placement in terms of logical volume fragments, for the logical volume containing the file. The **-l** and **-p** flags are mutually exclusive.

> **Note:** If neither the **-l** flag nor the**-p** flag is specified, the **-l** flag is implied by default. If both flags are specified, the **-p** flag is used.

-m *LogicalVolumeName* Displays the logical to physical map for a logical volume.

-p Displays file placement in terms of underlying physical volume, for the physical volumes that contain the file. If the logical volume containing the file is mirrored, the physical placement is displayed for each mirror copy. The **-l** and **-p** flags are mutually exclusive.

-v Displays more information about the file and its placement, including statistics on how widely the file is spread across the volume and the degree of fragmentation in the volume. The statistics are expressed in terms of either the logical or physical volume fragment numbers, depending on whether the **-l** or **-p** flag is specified.

File space efficiency is calculated as the number of nonnull fragments (*N*) divided by the range of fragments (*R*) assigned to the file and multiplied by 100, or (*N* /*R*) x 100. Range is calculated as the highest assigned address minus the lowest assigned address plus 1, or *MaxBlk*-*MinBlk*+1. For example, the logical blocks written for the file are 01550 through 01557, so *N* equals 8. The range, *R*, (01557 - 01550 +1) also equals 8. Space efficiency for this file is 100% or 8/8 x 100. The **-v** flag message prints the results of the (*N*/*R*)+100 equation.

According to this method of calculating efficiency, files greater than 32KB are never 100% efficient because of their use of the indirect block.

Sequential efficiency is defined as 1 minus the number of gaps (*nG*) divided by number of possible gaps (*nPG*) or 1 - (*nG*/*nPG*). The number of possible gaps equals *N* minus 1 (*nPG*=*N* - 1). If the file is written to 9 blocks (greater than 32KB), and the logical fragment column shows:

```
01550-01557
01600
```

The file is stored in 2 fragments out of a possible 9 fragments. The sequential efficiency calculation for this file is:

```
nG=1
nPG=9-1=8
(1-1/8) x 100=87.5%
```

Examples

1. To display the placement of a file in its logical volume, enter:
   ```
   fileplace data1
   ```
   This example displays the list of fragments and the logical volume that contains the file `data1`.
2. To display the indirect blocks for a file, enter:
   ```
   fileplace -i data1
   ```
   In addition to the default list of logical volume fragments, the indirect blocks (if any) used to store the file block addresses in the file system are enumerated.
3. To display more placement information for a file, enter:
   ```
   fileplace -v data1
   ```
   In addition to the default list of logical volume fragments, statistics about the placement efficiency are displayed.
4. To display all information about the placement of a file on its physical volumes, enter:
   ```
   fileplace -piv data1
   ```
   This example displays the list of file and indirect blocks in terms of the underlying physical volumes, and includes statistics about the efficiency of the placement.

Files

/dev/hd0, /dev/hd1, .../dev/hd*n*	Specifies the logical volume.

Related Information

The **sync** command.

Monitoring disk I/O in *Performance management*.

The Logical volume storage in*Operating system and device management* defines and discusses logical volume storage.

---

find Command

Purpose

Finds files with a matching expression.

Syntax

find [**-H** | **-L**] *Path ...* [*Expression*]

Description

The **find** command recursively searches the directory tree for each specified *Path* parameter, seeking files that match a Boolean expression written using the terms given in the following text. When the **find** command is recursively descending directory structures, it will not descend into directories that are symbolically linked into the current hierarchy. The output from the **find** command depends on the terms specified by the *Expression* parameter.

The **find** command does not support the 4.3 BSD fast find syntax.

Flags

-H Cause the file information and file type evaluated for each symbolic link encountered on the command line to be those of the file referenced by the link, and not the link itself. If the referenced file does not exist, the file information and type shall be for the link itself. File information for all symbolic links not on the command line shall be that of the link itself.

-L Cause the file information and file type evaluated for each symbolic link to be those of the file referenced by the link, and not the link itself.

Expression Terms

These Boolean expressions and variables describe the search boundaries of the **find** command as defined in the *Path* and *Expression* parameters.

Note: In the following definitions, the *n* variable specifies a decimal integer that can be expressed as **+***n* (more than *n*), **-***n* (less than *n*), or *n* (exactly *n*).

\(*Expression* **\)** Evaluates to the value True if the expression in parentheses is true.

-cpio *Device* Writes the current file to the specified device in the **cpio** command format.

-depth Always evaluates to the value True. Causes the descent of the directory hierarchy to be done so that all entries in a directory are affected before the directory itself is affected. This can be useful when the **find** command is used with the **cpio** command to transfer files that are contained in directories without write permission.

-ea Evaluates to the value True if file has either access control information (ACL) or extended attributes (EA) set.

-exec *Command* Evaluates to the value True if the specified command runs and returns a 0 value as exit status. The end of the specified command must be punctuated by a semicolon in quotation marks, an escaped semicolon, or a plus sign. An argument containing the two characters {} (braces) must be followed by a plus sign that punctuates the end of the specified command. A command parameter {} (braces) is replaced by the current path name.

-follow Causes symbolic and hard links to be followed.

-fstype *Type* Evaluates to the value True if the file system to which the file belongs is of the specified type, where the *Type* variable has a value of **jfs** (journaled file system) or **nfs** (network file system).

-group *Group* Evaluates to the value True if the file belongs to the specified group. If the value of the *Group* variable is numeric and does not appear in the **/etc/group** file, it is interpreted as a group ID.

-nogroup Evaluates to the value True if the file belongs to a group not in the **/etc/group** database.

-inum *n* Evaluates to the value True if file has an i-node matching the value of the *n* variable.

-links *n* Evaluates to the value True if the file has the specified number of links. See the **ln** command for a description of links.

-long Prints all available characters of each user/group name instead of truncating to the first 8 when used in combination with **-ls**.

-ls

Always evaluates to the value True. Causes the current path name to be printed together with its associated statistics. These statistics include the following:

- I-node number
- Size in kilobytes (1024 bytes)
- Protection mode
- Number of hard links
- User
- Group
- Size in bytes
- Modification time

If the file is a special file, the size field contains the major and minor device numbers. If the file is a symbolic link, the path name of the linked-to file is printed preceded by the -> (hyphen, greater than) symbols. Formatting is similar to that of the **ls -filds** command, however formatting is done internally without executing the **ls** command, therefore differences in output with the **ls** command may exist, such as with the protection mode.

-name *File*

Evaluates to the value True if the value of the *File* variable matches the file name. The usual shell filename generation characters (see the **sh** command) can be used. The pattern should either be enclosed in quotation marks or the escape character used when the **find** command is used from the shell. A backslash (\ ) is used as an escape character within the pattern. You can use wildcard (pattern-matching) characters, provided they are in quotation marks. See “Pattern matching with wildcards and metacharacters” in *Operating system and device management* for more information on using wildcard characters.

In an expression such as [a-z], the hyphen means *through* according to the current collating sequence. A collating sequence may define equivalence classes for use in character ranges. See “National Language Support Overview” in the *AIX Version 6.1 National Language Support Guide and Reference* for more information on collating sequences and equivalence classes.

-newer *File*

Evaluates to the value True if the current file has been modified more recently than the file indicated by the *File* variable.

-ok *Command*

The same as the **-exec** expression, except that the **find** command asks you whether it should start the specified command. An affirmative response starts the command. The end of the specified command must be punctuated by a semicolon enclosed in quotation marks or the \; (backslash-escape semicolon).

-perm [**-**] *OctalNumber*

Evaluates to the value True if the permission code of the file exactly matches the *OctalNumber* parameter (see the **chmod** command for an explanation of file permissions). If the optional **-** (hyphen) is present, this expression evaluates to true if at least these permissions are set. The *OctalNumber* parameter may be up to nine octal digits.

-perm [**-**] *Mode*

The mode argument is used to represent file mode bits. It will be identical in format to the <symbolicmode> operand described in **chmod**, and will be interpreted as follows:

Initially, a template will be assumed with all file mode bits cleared. Op symbols have the following function:

+ sets the appropriate mode bits in the template

- clears the appropriate bits

= sets the appropriate mode bits, without regard to the contents of the process' file mode creation mask

The op symbol - cannot be the first character of mode. This avoids ambiguity with the optional leading hyphen. Because the initial mode is all bits off, there are no symbolic modes that need to use - as the first character.

If the hyphen is omitted, the primary evaluates as True when the file permission bits exactly match the value of the resulting template. Otherwise, if mode is prefixed by a hyphen, the primary will evaluate as True if at least all bits in the resulting template are set in the file permission bits.

The *Mode* parameter is identical to the **chmod** command syntax. This expression evaluates to the value True if the file has exactly these permissions. If the optional **-** (hyphen) is present, this expression evaluates to the value True if at least these permissions are set.

-print Always evaluates to the value True. Displays the current path name. The **find** command assumes a **-print** expression, unless the **-exec, - ls**, or **-ok** expressions are present.

-prune Always evaluates to the value True. Stops the descent of the current path name if it is a directory. If the **-depth** flag is specified, the **-prune** flag is ignored.

-size *n* Evaluates to the value True if the file is the specified *n* of blocks long (512 bytes per block). The file size is rounded up to the nearest block for comparison.

-size *n***c** Evaluates to the value True if the file is exactly the specified *n* of bytes long. Adding **c** to the end of the *n* variable indicates that the size of the file is measured in individual bytes not blocks.

-atime *n* Evaluates as True if the file access time subtracted from the initialization time, divided by 86400 seconds (with any remainder discarded), is *n*. 86400 seconds is 24 hours.
Note: The definition of **-atime** has changed to comply with the Single UNIX Specification, Version 3. The previous behavior of **-atime** evaluated as True if the file had been accessed in *n*-1 to *n* multiples of 24 hours. By default, **find -atime** works like it did prior to UNIX03. The UNIX03 behavior can be obtained by setting the environment variables **XPG_SUS_ENV** to `ON` and **XPG_UNIX98** to `OFF`.

The previous behavior for this option can be obtained by setting the **XPG_UNIX98** variable to `ON`.

-ctime *n* Evaluates to the value True if the file i-node (status information) has been changed in the specified number of 24-hour periods.

-mtime *n*
Evaluates as True if the file modification time subtracted from the initialization time, divided by 86400 seconds (with any remainder discarded), is *n*. 86400 seconds is 24 hours.
Note: The definition of **-mtime** has changed to comply with the Single UNIX Specification, Version 3. The previous behavior of **-mtime** evaluated as True if the file had been modified in *n*-1 to *n* multiples of 24 hours. By default, **find -mtime** works like it did prior to UNIX03. The UNIX03 behavior can be obtained by setting the environment variables **XPG_SUS_ENV** to `ON` and **XPG_UNIX98** to `OFF`.

The previous behavior for this option can be obtained by setting the **XPG_UNIX98** variable to `ON`.

-amin *Number*
Evaluates to the value True if the file has been accessed in *Number*-1 to *Number* minutes. For example, **-amin 2** is true if the file has been accessed within 1 to 2 minutes.

-cmin *Number*
Evaluates to the value True if the file i-node (status information) has been changed in the specified number of minutes.

-mmin *Number*
Evaluates to the value True if the file has been modified in *Number*-1 to *Number* minutes

-type *Type*
Evaluates to the value True if the *Type* variable specifies one of the following values:

b Block special file

c Character special file

d Directory

f Plain file

l Symbolic link

p FIFO (a named pipe)

s Socket

-user *User*
Evaluates to the value True if the file belongs to the specified user. If the value of the *User* variable is numeric and does not appear as a login name in the **/etc/passwd** file, it is interpreted as a user ID.

-nouser
Evaluates to the value True if the file belongs to a user not in the **/etc/passwd** database.

-xdev
Always evaluates to the value True. Prevents the **find** command from traversing a file system different from the one specified by the *Path* parameter.

-xmode *Type*
Evaluates to the value True if the *Type* variable specifies one of the following values:

e The file is encrypted.

+ The file contains extended security information such as ACL, TCB, and so on.

E The file contains extended attributes information.

-slabel [+ | -]*sl*

Note: This attribute is valid only for Trusted AIX. For more information, see Trusted AIX in *Security*.

The **sl** argument is used to represent the sensitivity label (SL). The behaviors when the optional minus sign (-) or plus sign (+) is present are shown as follows:

+ The files that dominate (the same SL is not considered as domination) the given SL are returned.

- The files that are dominated (the same SL is not considered as domination) by the given SL are returned.

If the optional minus sign (-) or plus sign (+) is not present, files that have the same sensitivity label are returned.

-tlabel [+ | -]*tl*

Note: This attribute is valid only for Trusted AIX. For more information, see Trusted AIX in *Security*.

The **tl** argument is used to represent the integrity label (TL). The behaviors when the optional minus sign (-) or plus sign (+) is present are shown as follows:

+ The files that dominate (the same TL is not considered as domination) the given TL are returned.

- The files that are dominated (the same TL is not considered as domination) by the given TL are returned.

If the optional minus sign (-) or plus sign (+) is not present, files that have the same integrity label are returned.

-secflags *flag1[, flag2, ...]*

Note: This attribute is valid only for Trusted AIX. For more information, see Trusted AIX in *Security*.The **flag** argument is used to represent the File security flag (FSF) set as Trusted AIX system EA. The files that have the given FSF flag set are returned. If multiple flags are provided, files that have all given FSF flags set are returned.

-pdvirtual

Note: This attribute is valid only for Trusted AIX. For more information, see Trusted AIX in *Security*.

If a user having the aix.mls.pdir.mode authorization runs the **find** command, the command switches to real partitioned directory access mode and can access any hidden file, directory, and partitioned subdirectory.

By including this option, a user having the aix.mls.pdir.mode authorization can make the command switch to virtual partitioned directory access mode and run.

These expressions can be combined using the following operators in the order of decreasing precedence:

1. **(** *Expression* **)** - A parenthetic group of expressions and operators (parentheses are special to the shell and require the backslash-escape sequence).
2. **!** *Expression* - The negation of an expression (**'!'** is the unary NOT operator).
3. *Expression* [**-a**] *Expression* - Concatenation of expressions (the AND operation is implied by the juxtaposition of two primaries or may be explicitly stated as **-a**).
4. *Expression* **-o** *Expression* - Alternation of primaries; **-o** is the OR operator. The second expression will not be evaluated if the first expression is true.

Note: When using the **find** and **cpio** commands together, you must use the `-follow` option with the **cpio** command when using the `-L` option with the **cpio** command, and visa versa. Not using these two options together produces undesirable results. If no expression is present, `-print` as used in the default expression. For example, if the given expression does not contain any of the primaries

`-exec`, `-ok`, or `-print`, the given expression will be replaced by (*given_expression*) `-print`. The `-user`, `-group`, and `-newer` primaries each evaluate their respective arguments only once. Using a command specified by `-exec` or `-ok` does not affect subsequent primaries on the same file.

Security

aix.mls.pdir.mode	You must run the **find** command in real partitioned directory access mode.

Attention RBAC users and Trusted AIX users: This command can perform privileged operations. Only privileged users can run privileged operations. For more information about authorizations and privileges, see Privileged Command Database in *Security*. For a list of privileges and the authorizations associated with this command, see the **/etc/security/privcmds** database file.

Exit Status

This command returns the following exit values:

0	All *Path* parameters were traversed successfully.
>0	An error occurred.

Examples

1. To list all files in the file system with a given base file name, enter:

   ```
   find /  -name .profile -print
   ```

   This searches the entire file system and writes the complete path names of all files named **.profile**. The / (slash) tells the **find** command to search the root directory and all of its subdirectories. In order not to waste time, it is best to limit the search by specifying the directories where you think the files might be.
2. To list files having a specific permission code in the current directory tree, enter:

   ```
   find . -perm 0600  -print
   ```

   This lists the names of the files that have *only* owner-read and owner-write permission. The . (dot) tells the **find** command to search the current directory and its subdirectories. See the **chmod** command for an explanation of permission codes.
3. To search several directories for files with certain permission codes, enter:

   ```
   find manual clients proposals -perm -0600  -print
   ```

   This lists the names of the files that have owner-read and owner-write permission and possibly other permissions. The `manual`, `clients`, and `proposals` directories and their subdirectories are searched. In the previous example, `-perm 0600` selects only files with permission codes that match `0600` exactly. In this example, `-perm -0600` selects files with permission codes that allow the accesses indicated by `0600` and other accesses above the `0600` level. This also matches the permission codes 0622 and 2744.
4. To list all files in the current directory that have been changed during the current 24-hour period, enter:

   ```
   find .  -ctime 1  -print
   ```
5. To search for regular files with multiple links, enter:

   ```
   find .  -type f  -links +1  -print
   ```

This lists the names of the ordinary files (`-type f`) that have more than one link (`-links +1`).

Note: Every directory has at least two links: the entry in its parent directory and its own **.** (dot) entry. The **ln** command explains multiple file links.

6. To find all accessible files whose path name contains **find**, enter:

   ```
   find .  -name '*find*' -print
   ```

7. To remove all files named `a.out` or `*.o` that have not been accessed for a week and that are not mounted using **nfs**, enter:

   ```
   find / \( -name a.out -o -name '*.o' \) -atime +7 ! -fstype nfs -exec rm {} \;
   ```

   Note: The number used within the **-atime** expression is `+7`. This is the correct entry if you want the command to act on files not accessed for more than a week (seven 24-hour periods).

8. To print the path names of all files in or below the current directory, except the directories named `SCCS` or files in the `SCCS` directories, enter:

   ```
   find .  -name SCCS  -prune -o -print
   ```

   To print the path names of all files in or below the current directory, including the names of **SCCS** directories, enter:

   ```
   find . -print -name SCCS -prune
   ```

9. To search for all files that are exactly 414 bytes long, enter:

   ```
   find . -size 414c -print
   ```

10. To find and remove every file in your home directory with the **.c** suffix, enter:

    ```
    find /u/arnold -name "*.c" -exec rm {} \;
    ```

    Every time the **find** command identifies a file with the **.c** suffix, the **rm** command deletes that file. The **rm** command is the only parameter specified for the **-exec** expression. The {} (braces) represent the current path name.

11. In this example, `dirlink` is a symbolic link to the directory `dir`. You can list the files in `dir` by refering to the symbolic link `dirlink` on the command line. To do this, enter:

    ```
    find -H dirlink -print
    ```

12. In this example, `dirlink` is a symbolic link to the directory `dir`. To list the files in `dirlink`, traversing the file hierarchy under `dir` including any symbolic links, enter:

    ```
    find -L dirlink -print
    ```

13. To determine whether the file `dir1` referred by the symbolic link `dirlink` is newer than `dir2`, enter:

    ```
    find -H dirlink -newer dir2
    ```

    Note: Because the **-H** flag is used, time data is collected not from `dirlink` but instead from `dir1`, which is found by traversing the symbolic link.

14. To produce a listing of files in the current directory in **ls** format with expanded user and group name, enter :

    ```
    find . -ls -long
    ```

15. To list the files with ACL/EA set in current directory, enter:

    ```
    find . -ea
    ```

16. To list the files with SL equal to “IMPL_LO”, enter:

    ```
    find . –slabel IMPL_LO
    ```

17. To list the files with SL dominating “IMPL_LO”, enter:

    ```
    find . –slabel +IMPL_LO
    ```

18. To list the files with SL not dominating “TOP SECRET”, enter:

```
find . –slabel "-TOP SECRET"
```

19. To list the files with secflag FSF_PDIR, enter:
```
find . –secflags FSF_PDIR
```
20. To list the files with secflags FSF_AUDIT and FSF_TLIB, enter:
```
find . –secflags FSF_AUDIT,FSF_TLIB
```
21. For authorized users not to list the real path (including the partition subdirectory) under a partitioned directory, enter:
```
find . -pdvirtual
```

Files

/usr/bin/find	Contains the **find** command.
/bin/find	Symbolic link to the **find** command.
/etc/group	Contains a list of all known groups.
/etc/passwd	Contains a list of all known users.
/etc/security/enc/LabelEncodings	Contains the system labels.

Related Information

The **chmod** command, **cpio** command, **ln** command, **sh** command, **labck** command, **pdmkdir** command, **pdmode** command.

Backup methods in *Operating system and device management*.

Directories in *Operating system and device management*.

Types of files in *Operating system and device management*.

Input and output redirection in *Operating system and device management*.

Shells in *Operating system and device management*.

File and directory access modes in *Operating system and device management*.

Trusted AIX in *Security*.

---

finger Command

Purpose

Shows user information. This command is the same as the **f** command.

Syntax

{ **finger** | **f** }[[**-b**][**-h**] [**-l**][**-p**]]|[**-i**][**-q**][**-s**][**-w**]]

[**-f**][**-m**][*User*| *User* **@***Host*| **@***Host*]

Description

The **/usr/bin/finger** command displays information about the users currently logged in to a host. The format of the output varies with the options for the information presented.

Default Format

The default format includes the following items:

- Login name
- Full user name
- Terminal name
- Write status (an * (asterisk) before the terminal name indicates that write permission is denied)

For each user on the host, the default information list also includes, if known, the following items:

- Idle time (Idle time is minutes if it is a single integer, hours and minutes if a : (colon) is present, or days and hours if a ″d″ is present.)
- Login time
- Site-specific information

The site-specific information is retrieved from the gecos field in the **/etc/passwd** file. The gecos field may contain the Full user name followed by a comma or / (slash character). All information that follows the comma or slash character is displayed by the finger command with the Site-specific information.

Longer Format

A longer format is used by the **finger** command whenever a list of user's names is given. (Account names as well as first and last names of users are accepted.) This format is multiline, and includes all the information described above along with the following:

- User's **$HOME** directory
- User's login shell
- Contents of the **.plan** file in the user's **$HOME** directory
- Contents of the **.project** file in the user's **$HOME** directory

The **finger** command may also be used to look up users on a remote system. The format is to specify the user as *User@Host*. If you omit the user name, the **finger** command provides the standard format listing on the remote system.

Create the **.plan** and **.project** files using your favorite text editor and place the files in your **$HOME** directory. The **finger** command uses the **toascii** subroutine to convert characters outside the normal ASCII character range when displaying the contents of the **.plan** and **.project** files. The **finger** command displays a `M-` before each converted character.

When you specify users with the *User* parameter, you can specify either the user's first name, last name, or account name. When you specify users, the **finger** command, at the specified host, returns information about those users only in long format.

For other information about the **finger** command, see ″Installation of TCP/IP″ in *Networks and communication management*.

Flags

-b Gives a brief, long-form listing.
-f Suppresses printing of header line on output (the first line that defines the fields that are being displayed).
-h Suppresses printing of **.project** files on long and brief long formats.
-i Gives a quick listing with idle times.
-l Gives a long-form listing.
-m Assumes that the *User* parameter specifies a user ID (used for discretionary access control), *not* a user login name.
-p Suppresses printing of **.plan** files on long-form and brief long-form formats.
-q Gives a quick listing.

-s Gives a short format list.
-w Gives a narrow, short-format list.

Parameters

@Host	Specifies all logged-in users on the remote host.
User	Specifies a local user ID (used for discretionary access control) or local user login name, as specified in the **/etc/passwd** file.
User@Host	Specifies a user ID on the remote host, displayed in long format.

Examples

1. To get information about all users logged in to host `alcatraz`, enter:

   ```
   finger @alcatraz
   ```

   Information similar to the following is displayed:

   ```
   [alcatraz.austin.ibm.com]
   Login     Name        TTY Idle         When      Site Info
   brown    Bob Brown   console   2d   Mar 15 13:19
   smith    Susan Smith    pts0  11:   Mar 15 13:01
   jones    Joe Jones      tty0    3   Mar 15 13:01
   ```

   User `brown` is logged in at the `console`, user `smith` is logged in from pseudo teletype line `pts0`, and user `jones` is logged in from `tty0`.

2. To get information about user `brown` at `alcatraz`, enter:

   ```
   finger brown@alcatraz
   ```

   Information similar to the following is displayed:

   ```
   Login name: brown
   Directory: /home/brown    Shell: /home/bin/xinit -L -n Startup
   On since May 8 07:13:49 on console
   No Plan.
   ```

3. To get information about user `brown` at a local host in short form, enter:

   ```
   finger -q brown
   ```

   Information similar to the following is displayed:

   ```
   Login           TTY             When
   brown           pts/6           Mon Dec1710:58
   ```

Files

/usr/bin/finger	Contains the **finger** command.
/etc/utmp	Contains list of users currently logged in.
/etc/passwd	Defines user accounts, names, and home directories.
/etc/security/passwd	Defines user passwords.
/var/adm/lastlog	Contains last login times.
$HOME/.plan	Optional file that contains a one-line description of a user's plan.
$HOME/.project	Optional file that contains a user's project assignment.

Related Information

The **hostname** command, **rwho** command.

The **fingerd** daemon.

Command for displaying information about logged-in users in *Networks and communication management*.

Communications and networks in *Networks and communication management*.

---

fingerd Daemon

Purpose

Provides server function for the **finger** command.

Syntax

Note: The **fingerd** daemon is usually started by the **inetd** daemon. It can also be controlled from the command line, using System Resource Controller (SRC) commands.

/usr/sbin/fingerd [**-s**] [**-f**]

Description

The **/usr/sbin/fingerd** daemon is a simple protocol that provides an interface to the **finger** command at several network sites. The **finger** command returns a status report on either the current system or a user. The **fingerd** daemon listens for Transmission Control Protocol (TCP) requests at port 79 as listed in the **/etc/services** file and the /**etc**/**inetd.conf** file.

For individual site security concern the **fingerd** daemon, by default, will not forward any **finger** request to any other system. If it receives a **finger** forward request, the **fingerd** daemon replies with the message `Finger forwarding service denied` to the **finger** command. The system administractor has the option to turn on finger forwarding as the default when running the **fingerd** daemon by using the **-f** flag.

Changes to the **fingerd** daemon can be made using the System Management Interface Tool (SMIT) or SRC or by editing the **/etc/inetd.conf** file or /**etc**/**services** file. Entering `fingerd` at the command line is not recommended. The **fingerd** daemon is started by default when it is uncommented in the **/etc/inetd.conf** file.

The **inetd** daemon get its information from the /**etc**/**inetd.conf** file and the /**etc**/**services** file.

After changing the /**etc**/**inetd.conf** or /**etc**/**services** file, run the **refresh -s inetd** or **kill-1***InetdPID* command to inform the **inetd** daemon of the changes to its configuration file.

The **fingerd** daemon should have a user ID with the least privileges possible. The **nobody** ID allows the least permissions. Giving the **fingerd** daemon the **nobody** user ID allows the daemon to be used on your host. Change the /**etc**/**services** file to the reflect the user ID you want to use.

Manipulating the fingerd Daemon with the System Resource Controller

The **fingerd** daemon is a subserver of the **inetd** daemon, which is a subsystem of the SRC. The **fingerd** daemon is a member of the **tcpip** SRC subsystem group. This daemon is enabled when it is uncommented in the **/etc/inetd.conf** file and can be manipulated by the following SRC commands:

startsrc	Starts a subsystem, group of subsystems, or a subserver.
stopsrc	Stops a subsystem, group of subsystems, or a subserver.
lssrc	Gets the status or a subsystem, group or subsystems, or a subserver.

Flags

-s Turns on socket-level debugging.

-f Turns on finger forwarding service for this **fingerd** daemon.

Examples

Note: The arguments for the **fingerd** daemon can be specified by using SMIT or by editing the **/etc/inetd.conf** file.

1. To start the **fingerd** daemon type:

   ```
   startsrc -t finger
   ```

   This command starts the **fingerd** subserver.
2. To stop the **fingerd** daemon usually, type:

   ```
   stopsrc -t finger
   ```

   This command allows all pending connections to start and existing connections to complete but prevents new connections from starting.
3. To force stop the **fingerd** daemon and all **fingerd** connections type:

   ```
   stopsrc -t -f finger
   ```

   This command terminates all pending connections and existing connections immediately.
4. To display a short status report about the **fingerd** daemon type:

   ```
   lssrc -t finger
   ```

   This command returns the daemon's name, process ID, and state (active or inactive).

Related Information

The **finger** command, **lssrc** command, **kill** command, **refresh** command, **startsrc** command, **stopsrc** command.

TCP/IP daemons in *Networks and communication management*.

The **/etc/inetd.conf** file format, **/etc/services** file format.

For information on installing the Web-based System Manager, see Chapter 2: Installation and System Requirements in *AIX Version 6.1 Web-based System Manager Administration Guide*.

---

fish Command

Purpose

Plays the go fish card game.

Syntax

fish

Description

The object of the go fish game is to accumulate books of four cards with the same face value. You and the program (your opponent) take turns asking for cards from one another's hand. If your opponent has one or more cards of the value requested, your opponent must hand them over. If not, your opponent prompts `GO FISH!`, and you draw a card from the pool of undealt cards. If you draw the card you asked for, you draw again. As books are made, they are laid down on the table. Play continues until there are no cards left. The player with the most books wins the game. The **fish** command tells you the winner and exits.

The **fish** command prompts with `instructions?` before play begins. To see the instructions, enter `Y` (yes).

Entering a `p` as your first move gives you the professional-level game. The default is an amateur-level game.

When playing go fish, you enter the card you want when your opponent prompts:

```
you ask me for:
```

If you press only the Enter key when prompted, you receive information about the number of cards in your opponent's hand and in the pool.

The game displays:

- your current hand, including the books you have accumulated
- `GO FISH!` when either you or your opponent ask for a card the other does not have
- the card drawn after the `GO FISH!` prompt
- the card your opponent asks you for
- completed books (yours or your opponent's)
- the requested card when you or your opponent get another guess.

Examples

The following is a sample of a **fish** screen display:

```
your hand is: A 5 5 7 10 J Q
you ask me for: 5
I say "GO FISH!"
You draw A
I ask you for: 5
Made a book of 5's
I get another guess
I ask you for 6
You say "GO FISH!"
your hand is: A A 7 10 J Q
you ask me for:
```

To exit the game before play is completed, press the Interrupt (Ctrl-C) key sequence.

Files

/usr/games Location of the system's games.

Related Information

The **arithmetic** command, **back** command, **bj** command, **craps** command, **fortune** command, **hangman** command, **moo** command, **number** command, **quiz** command, **ttt** command, **turnoff** command, **turnon** command, **wump** command.

---

flcopy Command

Purpose

Copies to and from diskettes.

Syntax

flcopy [**-f** *Device*] [**-h** | **-r**] [**-t** *Number*]

Description

The **flcopy** command copies a diskette (opened as **/dev/rfd0**) to a file named **floppy** created in the current directory, then prints the message: `Change floppy, hit return when done`. The **flcopy** command then copies the **floppy** file to the diskette. You can specify the **-f**, **-h**, **-r**, or **-t***Number* flag to modify the behavior of the **flcopy** command.

> **Note:** You cannot use the **flcopy** command to copy data from one diskette to another diskette of different size.

Flags

-f *Device*	Allows you to specify a drive other than **/dev/rfd0**.
-h	Causes the **flcopy** command to open the **floppy** file in the current directory and copy it to **/dev/rfd0**.
-r	Tells the **flcopy** command to exit after copying the diskette to the **floppy** file in the current directory.
-t *Number*	Causes only the specified *Number* of tracks to be copied. The tracks copied always begin with the first tracks on the diskette.

Exit Status

This command returns the following exit values:

0	Successful completion.
>0	An error occurred.

Examples

1. To copy **/dev/rfd1** to the **floppy** file in the current directory, enter:

   ```
   flcopy -f/dev/rfd1 -r
   ```

2. To copy the first 100 tracks of the diskette, enter:

   ```
   flcopy -f/dev/rfd1 -t100
   ```

Files

/usr/sbin/flcopy	Contains the **flcopy** command.

Related Information

The **format** or **fdformat** command.

The **fd** special file.

---

flush-secldapclntd Command

Purpose

The **flush-secldapclntd** command flushes the cache for the **secldapclntd** daemon process.

Syntax

//usr/sbin/flush-secldapclntd

Description

The **flush-secldapclntd** command clears the cache for the **secldapclntd** daemon process.

Example

1. To flush the **secldapclntd** daemon cache, type:

```
/usr/sbin/flush-secldapclntd
```

Files

/etc/security/ldap/ldap.cfg	Contains information needed by the **secldapclntd** daemon to connect to the server.

Related Information

The **secldapclntd** daemon

The **mksecldap**, **stop-secldapclntd**, **start-secldapclntd**, **restart-secldapclntd**, and **ls-secldapclntd** commands.

The **/etc/security/ldap/ldap.cfg** file.

---

fmt Command

Purpose

Formats mail messages prior to sending.

Syntax

/usr/bin/fmt [**-***Width*] [*File* ...]

Description

The **fmt** command starts a text formatter that reads the concatenation of input *Files* (or standard input if no *Files* are specified), then produces on standard output a version of the input with the line lengths set to the value of **-***Width*. If no value is specified with the **-***Width* flag, the default value of 72 characters is used. The spacing at the beginning of the input lines is preserved in the output, as are blank lines and spacing between words.

The **fmt** command is generally used to format mail messages to improve their appearance before they are sent. However, the **fmt** command may also be useful for simple formatting tasks. For example, within visual mode of a text editing program such as the vi editor, the command **!}fmt** formats a paragraph so that all lines are set to the value specified with the **-***Width* flag. If no value is specified with the -*Width* flag, the default value of 72 characters is used. Standard text editing programs are more appropriate than **fmt** for complex formatting operations.

> **Note:** Do not use the **fmt** command if the message contains embedded messages or preformatted information from other files. This command formats the heading information in embedded messages and may change the format of preformatted information.

Flags

File	Specifies the name of the file to be formatted.
-*Width*	Specifies the line length. The default value for *Width* is 72 characters.

Examples

1. To format a message you have created with the mail editor, enter:

   ```
   ~| fmt
   ```

   The ~| is entered at the left margin of the message. After you issue the ~| `fmt` command, the message is formatted. The word (`continue`) is displayed to indicate that you can enter more information or send the message.
2. To format a file and display the output on your screen, enter:

   ```
   fmt file1
   ```

   In this example, the file `file1` is formatted and displayed on your screen.

Files

/usr/bin/fmt	Contains the **fmt** command.

Related Information

The **mail** command, **nroff** command, **vi** command.

Mail applications in *Networks and communication management*.

---

fold Command

Purpose

Folds long lines for fixed-width output devices.

Syntax

fold [**-b**] [**-s**] [**-w** *Width*] [*File...*]

Description

The **fold** command is a filter that folds long lines for a finite-width output device. By default, the command folds the contents of standard input, breaking the lines to a line width of 80 (eighty). You can also specify one or more files as input to the command.

The **fold** command inserts a new-line character in the input lines so that each output line is as wide as possible without exceeding the value specified by the *Width* parameter. If the **-b** flag is specified, line width is counted in bytes. If the **-b** flag is not specified:

- *Width* is counted in columns as determined by the **LC_CTYPE** environment variable.
- A backspace character decreases the length of an output line by 1.
- A tab character advances to the next column where the column position is 1 plus a multiple of 8.

The **fold** command accepts **-w** *Width* values in multiples of 8 if the file contains tabs. To use other width values when the file contains tabs, use the **expand** command before using the **fold** command.

Notes:

1. The **fold** command may affect any underlining that is present.
2. The **fold** command does not insert new-line characters in the middle of multibyte characters even when the **-b** flag is used.

Flags

-b	Counts *Width* in bytes. The default is to count in columns.
-s	Breaks the line after the rightmost blank within the *Width* limit, if an output line segment contains any blank characters. The default is to break lines so each output line segment is as wide as possible.
-w *Width*	Specifies the maximum line width as the value of the *Width* variable. The default is 80.

Exit Status

This command returns the following exit values:

0	All input files processed successfully.
>0	An error occurred.

Examples

To fold the lines of a file named `longlines` into width 72 (seventy-two), enter:

```
fold -w 72 longlines
```

Files

/usr/bin/fold	Contains the **fold** command.

Related Information

The **expand** command, **tab** command.

---

folder Command

Purpose

Selects and lists folders and messages.

Syntax

folder [**+** *Folder*] [*Message*] [**-all**] [**-nopack** | **-pack**] [**-nofast** | **-fast**] [**-norecurse** | **-recurse**] [**-print** | **-noprint**] [**-header** | **-noheader**] [**-nototal** | **-total**] [**-push** | **-pop**] [**-list** | **-nolist**]

Description

The **folder** command sets the current folder and the current message for that folder, and lists information about your folders. By default, the **folder** command lists the current folder name, the number of messages, the range of the message numbers, and the current message.

The folder specified by the **+***Folder* flag becomes the current folder. The message specified by the *Message* parameter becomes the current message for the folder. Use the **-pack** flag to renumber the messages in a folder.

Flags

-all	Displays a line of information about each folder in your mail directory.
-fast	Displays only the names of the folders.
+*Folder*	Specifies the folder information to display.
-header	Displays column headings for the folder information.

-help Lists the command syntax, available switches (toggles), and version information.

Note: For Message Handler (MH), the name of this flag must be fully spelled out.

-list Displays the current folder followed by the contents of the folder stack.

Message Sets the specified message as the current message. Unless you specify the **+***Folder* flag, the command sets the specified message for the current folder. Use the following references to specify a message:

Number
Number of the message.

cur or . (period)
Current message. This is the default.

first First message in a folder.

last Last message in a folder.

next Message following the current message.

new The new message that is created.

prev Message preceding the current message.

-nofast Displays information about each folder. This flag is the default.

-noheader Suppresses column headings for the folder information. This flag is the default.

-nolist Suppresses the display of the folder-stack contents. This flag is the default.

-nopack Prevents renumbering of the messages in the folder. This flag is the default.

-noprint Prevents display of folder information. If the **-push**, **-pop**, or **-list** flag is specified, the **-noprint** flag is the default.

-norecurse Displays information about the top-level folders in your current folder only. Information about subfolders is not displayed. This flag is the default.

-nototal Prevents display of the total of all messages and folders in your mail directory structure. When the **-all** flag is specified, the default is the **-total** flag; otherwise, the **-nototal** flag is the default.

-pack Renumbers the messages in the specified folder. Renumbering eliminates gaps in the message numbering after messages have been deleted.

-pop Removes the folder from the top of the folder stack and makes it the current folder. The **+***Folder* flag cannot be specified with the **-pop** flag.

-print Displays information about the folders. If the **-push**, **-pop**, or **-list** flag is specified, the **-noprint** flag is the default; otherwise, the **-print** flag is the default.

-push Moves the current folder to the top of the folder stack and sets the specified folder as the current folder. If no folder is specified, the **-push** flag swaps the current folder for the folder on top of the folder stack.

-recurse Displays information about all folders and subfolders in your current folder.

-total Displays all messages and folders in your mail directory structure. The **-total** flag does not display information for subfolders unless you specify the **-recurse** flag. The **-total** flag is the default if the **-all** flag is specified.

Profile Entries

The following entries are entered in the *UserMhDirectory*/**.mh_profile** file:

`Current-Folder:` Sets the default current folder.

`Folder-Protect:` Sets the protection level for the new folder directories.

`Folder-Stack:` Specifies the folder stack.

`lsproc:` Specifies the program used to list the contents of a folder.

`Path:` Specifies the user's MH directory.

Security

Attention RBAC users and Trusted AIX users: This command can perform privileged operations. Only privileged users can run privileged operations. For more information about authorizations and privileges,

see Privileged Command Database in *Security*. For a list of privileges and the authorizations associated with this command, see the **/etc/security/privcmds** database file.

Examples

1. To display information about the current folder, enter:

   ```
   folder
   ```

   The system responds with a message similar to the following:

   ```
   inbox+ has 80 messages (1-82); cur = 7; (others).
   ```

   In this example, the current folder is `inbox`.The folder contains `80` `messages`, ranging from message `1` to message `82`. The current message number is `7`.

2. To display information about all folders, enter:

   ```
   folder -all
   ```

   The system responds with a message similar to the following:

   ```
   Folder    # of messages (range); cur msg    (other files)
   inbox+ has 80 messages (1-82); cur= 7; (others).
   test    has    5 messages (1-5);    cur= 5; (others).

           Total= 85 messages in 2 folders
   ```

   In this example, there are `2` folders containing a total of `85` messages. The current folder is `inbox`, indicated by the + (plus sign) that follows it.

3. To make the `test` folder the current folder and display information about `test`, enter:

   ```
   folder +test
   ```

   The system responds with a message similar to the following:

   ```
   test+ has 5 messages (1-5); cur = 5; (others)
   ```

4. To make message `2` the current message in the current folder, enter:

   ```
   folder 2
   ```

   The system responds with a message similar to the following:

   ```
   test+ has 5 messages (1-5); cur = 2; (others)
   ```

5. To create a folder called `group` and make it the current folder, enter:

   ```
   folder +group
   ```

   The system responds with a message similar to the following:

   ```
   Create folder "/home/dawn/Mail/group"? _
   ```

   Enter:

   ```
   yes
   ```

   The system responds with a message similar to the following:

   ```
   group+ has no messages.
   ```

6. To renumber the messages in the current folder, enter:

   ```
   folder -pack
   ```

   The system responds with a message similar to the following:

```
inbox+  has  80  messages  (1-80);  cur=  7;  (others).
```

In this example, the messages are renumbered to eliminate gaps in the message numbering after messages have been deleted.

Files

$HOME/.mh_profile	Contains the MH user profile.
/usr/bin/folder	Contains the **folder** command.

Related Information

The **folders** command, **mhpath** command, **packf** command, **refile** command.

The **mh_alias** file format, **mh_profile** file format.

Mail applications in *Networks and communication management*.

---

folders Command

Purpose

Lists all folders and messages in mail directory.

Syntax

folders [**+***Folder*] [*Message*] [**-all**] [**-pack** | **-nopack**] [**-fast** | **-nofast**] [**-recurse** | **-norecurse**] [**-print** | **-noprint**] [**-header** | **-noheader**] [**-total** | **-nototal**] [**-push** | **-pop**] [**-list** | **-nolist**]

Description

The **folders** command lists all folders and messages in your mail directory. This command is equivalent to the **folder** command specified with the **-all** flag.

Flags

-all	Displays a line of information about each folder in your mail directory.
-fast	Displays only the names of the folders.
+*Folder*	Specifies the folder information to display.
-header	Displays column headings for the folder information. This flag is the default.
-help	Lists the command syntax, available switches (toggles), and version information. **Note:** For Message Handler (MH), the name of this flag must be fully spelled out.
-list	Displays the current folder followed by the contents of the folder stack.

Message Sets the specified message as the current message. Unless you specify the **+***Folder* flag, the command sets the specified message for the current folder. Use the following references to specify a message:

Number
Number of the message.

cur or . (period)
Current message. This is the default.

first First message in a folder.

last Last message in a folder.

next Message following the current message.

new The new message that is created.

prev Message preceding the current message.

-nofast Displays information about each folder. This flag is the default.

-noheader Suppresses column headings for the folder information.

-nolist Suppresses the display of the folder-stack contents. This flag is the default.

-nopack Prevents renumbering of the messages in the folder. This flag is the default.

-noprint Prevents display of folder information. If the **-push**, **-pop**, or **-list** flag is specified, the **-noprint** flag is the default.

-norecurse Displays information about the folders in your mail directory. Information about subfolders is not displayed. This flag is the default.

-nototal Prevents display all messages and folders in your mail directory structure.

-pack Renumbers the messages in the folders. Renumbering eliminates gaps in message numbering after messages have been deleted.

-pop Removes the folder from the top of the folder stack and makes it the current folder.

-print Displays the number of messages in each folder, the current message for each folder, and the current folder. If the **-push**, **-pop**, or **-list** flag is specified, the **-noprint** flag is the default; otherwise, the **-print** flag is the default.

-push Moves the current folder to the top of the folder stack and sets the specified folder as the current folder. If no folder is specified, the **-push** flag swaps the current folder for the folder on top of the folder stack.

-recurse Displays information about all folders and subfolders in your mail directory structure.

-total Displays all messages and folders in your mail directory structure. The **-total** flag does not display information for subfolders unless you specify the **-recurse** flag. The **-total** flag is the default.

Profile Entries

The following entries are entered in the *UserMhDirectory*/**.mh_profile** file:

`Current-Folder:` Sets the default current folder.

`Folder-Protect:` Sets the protection level for the new folder directories.

`Folder-Stack:` Specifies the folder stack.

`lsproc:` Specifies the program used to list the contents of a folder.

`Path:` Specifies the user's MH directory.

Examples

1. To display information about all folders, enter:

   ```
   folders
   ```

   The system responds with a message similar to the following:

```
Folder  # of messages (range); cur msg  (other files)
inbox+ has 80 messages (1-82); cur= 7; (others).
test has 5 messages (1-6);   cur=  5; (others).

   Total= 85 messages in 2 folders.
```

In this example, there are 2 folders containing a total of 85 messages. The current folder is inbox, indicated by the + (plus sign) following it.

2. To list only the names of all folders, enter:

```
folders    -fast
```

The system responds with a message similar to the following:

```
inbox
test
```

3. To renumber the messages in all folders, enter:

```
folders    -pack
```

The system responds with a message similar to the following:

```
inbox+ has 80 messages (1-80); cur= 7; (others).
test has 5 messages (1-5);  cur= 5; (others).
```

In this example, the messages in the inbox folder and in the test folder have been renumbered to eliminate gaps in message numbering after messages were deleted.

Files

$HOME/.mh_profile	Contains the MH user profile.
/usr/bin/folders	Contains the **folders** command.

Related Information

The **folder** command, **mhpath** command, **packf** command, **refile** command.

The **mh_alias** file format, **mh_profile** file format.

Mail applications in *Networks and communication management*.

---

format Command

Purpose

Formats either diskettes or read/write optical media disks.

Syntax

format [**-d** *Device*] [**-f**] [**-l**]

Description

Attention: Formatting a diskette or read/write optical disk destroys any existing data on it.

The **format** command formats diskettes in the diskette drive specified by the *Device* parameter. The **format** command determines the device type, which may be one of the following:

- 5.25-inch low-density diskette (360KB) containing 40x2 tracks, each with 9 sectors

- 5.25-inch high-capacity diskette (1.2MB) containing 80x2 tracks, each with 15 sectors
- 3.5-inch low-density diskette (720KB) containing 80x2 tracks, each with 9 sectors
- 3.5-inch high-capacity diskette (1.44MB) containing 80x2 tracks, each with 18 sectors
- 3.5-inch high-capacity diskette (2.88MB) containing 80x2 tracks, each with 36 sectors

The sector size is 512 bytes for all diskette types.

The **format** command formats a diskette with the highest capacity supported by the diskette drive, unless the *Device* parameter specifies a different density.

The **format** command formats a read/write optical disk, provided that the drive supports setting the Format Options Valid (FOV) bit of the defect list header to 0. To format a read/write optical disk, use the name of the read/write optical drive (such as **/dev/romd0**) after the **-d** flag. For more information, see the **DKFORMAT** operation of the **ioctl** subroutine in ″scdisk SCSI Device Driver″ in *AIX Version 6.1 Technical Reference: Kernel and Subsystems Volume 2*.

Before formatting a diskette or read/write optical disk, the **format** command prompts for verification. This allows you to end the operation cleanly.

Flags

-d *Device* Specifies the device used to format the diskette. If the device name ends with the letter **h**, the drive formats the diskette for high density. If the device name ends with the letter **l**, the drive formats the diskette for low density. Refer to the **fd** special file for information about valid device types. This flag is used only with the **format** command.

> **Attention:** If the diskette drive supports a higher capacity than the highest capacity for which the diskette was manufactured, the capacity of the diskette should be explicitly stated in the *Device* parameter (**-d** *Device* flag) of the **format** command. For example, to format a 1MB diskette on a 4MB diskette drive, specify the diskette capacity in the **-d** flag as follows:
>
> ```
> -d /dev/fd0.9 for a 1MB diskette
> ```
>
> Failure to do this may cause read and write errors.

-f Formats the diskette without checking for bad tracks, thus formatting the diskette more quickly. This flag applies to diskettes only, not to read/write optical disks. It is used only with the **format** command.

-l (Lowercase L) Formats a 360KB diskette in a 5.25-inch, 1.2MB diskette drive. Formats a 720KB diskette in a 3.5-inch 1.4MB diskette drive. This flag applies to diskettes only, not to read/write optical disks. It is used only with the **format** command.

> **Attention:** A 360KB diskette drive may not be able to read a 360KB diskette that has been formatted in a 1.2MB drive.

Parameters

Device Specifies the device containing the diskette to be formatted. The default is the **/dev/rfd0** device for drive 0.

Examples

1. To format a diskette in the **/dev/rfd0** device, enter:
   ```
   format   -d   /dev/rfd0
   ```
2. To format a diskette without checking for bad tracks, enter:
   ```
   format   -f
   ```

3. To format a 360KB diskette in a 5.25-inch, 1.2MB diskette drive in the **/dev/rfd1** device, enter:
   ```
   format -l -d /dev/rfd1
   ```
4. To format a 3.5-inch, low-density (720KB) diskette, enter:
   ```
   format -d /dev/fd0.9
   ```
5. To format a 3.5-inch, high-capacity (1.44MB) diskette, enter:
   ```
   format -d /dev/fd0.18
   ```
6. To format a read/write optical disk in the **/dev/romd0** device, enter:
   ```
   format -d /dev/romd0
   ```

Files

/usr/sbin/format	Contains the **format** command.
/dev/rfd*	Specifies the device parameters.
/dev/fd*	Specifies the device parameters.
/dev/romd*	Specifies the device parameters.
/dev/omd*	Specifies the device parameters.

Related Information

The **flcopy** command, **fdformat** command.

The **fd** special file.

---

fortune Command

Purpose

Displays a random fortune from a database of fortunes.

Syntax

fortune [**-**] [**-s** | **-l** | **-a** [**-w**]] [*File*]

Description

The **fortune** command displays a fortune from either the **fortunes.dat** file or the file specified by the *File* parameter. After displaying the fortune, the **fortune** command exits.

Flags

-	Displays the usage summary.
-a	Displays either type of fortune.
-l	Displays long fortunes only.
-s	Displays short fortunes only.
-w	Waits after displaying a fortune to allow the user time to read the fortune.

Files

/usr/games	Location of the system's games.
/usr/games/lib/fortune/fortunes.dat	Location of the default **fortune** database.

Related Information

The **arithmetic** command, **back** command, **bj** command, **craps** command, **fish** command, **hangman** command, **moo** command, **number** command, **quiz** command, **ttt** command, **turnoff** command, **turnon** command, **wump** command.

---

forw Command

Purpose

Forwards messages.

Syntax

forw [**+** *Folder*] [**-draftfolder +***Folder* | **-nodraftfolder**] [*Message*] [**-draftmessage** *Message*] [**-digest** *Name* [**-issue** *Number*] [**-volume** *Number*]] [**-form** *FormFile*] [**-editor** *Editor* | **-noedit**] [**-whatnowproc** *Program* | **-nowhatnowproc**] [**-filter***File*] [**-annotate** [**-inplace** | **-noinplace**] | **-noannotate**] [**-format** | **-noformat**] [**-help**]

Description

The **forw** command starts an interface for forwarding messages. By default, the **forw** command interface:

- Opens for editing a *UserMhDirectory***/draft** file.
- Prompts the user to enter forwarding information based on the template defined in the **/etc/mh/mhl.forward** file.
- Prompts the user to enter any additional text that should accompany the forwarded message.

To complete editing of the *UserMhDirectory***/draft** file, press the Ctrl-D sequence. The **forw** command appends the current message from the current folder to the **draft** file. If you want to append more than one message, use the *Messages* parameter.

> **Note:** A line of dashes or a blank line must be left between the header and the body of the message for the message to be identified when it is sent.

Upon exiting the editor, the **forw** command starts the `What Now?` prompt. Press the Enter key to see a list of the available **whatnow** subcommands. These subcommands enable you to continue to edit the message, list the message, direct the disposition of the message, or end the processing of the **forw** command.

The **forw** command allows you to change the format of the forwarded message with the **-form** flag. By default, the command uses the default message format located in your *UserMhDirectory***/forwcomps** file. If you have not defined your own **forwcomps** file, the **/etc/mh/forwcomps** file is used.

Use the **-annotate** flag to annotate the original message with forwarding information. To ensure annotation, send the forwarded note before exiting the **forw** command interface.

> **Note:** The **-annotate** flag is not preserved over multiple executions of the **forw** command on the same draft.

Flags

-annotate Annotates the forwarded messages with the lines:

```
Forwarded: Date
Forwarded: Addresses
```

Use the **-inplace** flag to force annotation in place. This preserves links to the annotated message.

-digest *Name*
Uses the digest facility to create a new issue for the digest specified by the *Name* variable. The **forw** command expands the format strings in the **components** file (using the same format string mechanism used by the **repl** command) and composes the draft using the standard digest encapsulation algorithm. After the draft has been composed, the **forw** command writes out the volume and issue entries for the digest and starts the editor.

Unless you specify the **-form** flag, the **forw** command uses the format in the *UserMhDirectory***/digestcomps** file. If this file does not exist, the command uses the default specified in the **/etc/mh/digestcomps** file.

-draftfolder +*Folder*
Places the draft message in the specified folder. If you do not specify this flag, the **forw** command selects a default draft folder according to the information supplied in the Message Handler (MH) profiles. If **+***Folder* is not specified, the `Current-Folder` is assumed. You can define a default draft folder in the **$HOME/.mh_profile** file.

Note: If **-draftfolder +***Folder* is followed by a *Message* parameter, it is the same as specifying the **-draftmessage** flag.

-draftmessage *Message*
Identifies a draft message. If you specify **-draftfolder** without the **-draftmessage** flag, then the default message is `new`.

-editor *Editor*
Specifies the initial editor for preparing the message.

-filter *File*
Reformats each message being forwarded and places the reformatted message in the draft message. The **-filter** flag accepts formats used by the **mhl** command.

+*Folder*
Specifies the folder that contains the messages you want to forward. If a folder is not specified, `Current-Folder` is assumed.

-form *FormFile*
Displays the **forw** command output in the format specified by the *FormFile* variable. The **forw** command treats each line in the specified file as a format string. If the **-digest** flag is also specified, the **forw** command uses the form specified by the *File* variable as the format of the digest. If the **-form** flag is not specified when the **-digest** flag is used, the digest filter file becomes the form default.

-format
Using the **mhl** command and a default format file, reformats each message being forwarded and places the reformatted message in the draft message. If the *UserMhDirectory***/mhl.forward** file exists, it contains the default format. Otherwise, the **/etc/mh/mhl.forward** file contains the default format.

-help
Lists the command syntax, available switches (toggles), and version information.

Note: For MH, the name of this flag must be fully spelled out.

-inplace
Forces annotation to be done in place to preserve links to the annotated message.

-issue *Number*
Specifies the issue number of the digest. The default issue number is one greater than the current value of the `DigestName-issue-list` entry in the *UserMhDirectory***/context** file.

Message
Specifies a message. You can specify several messages, a range of messages, or a single message. Use the following references when specifying messages:

Number
Number of the message.

Sequence
A group of messages specified by the user. Recognized values include:

all All messages in the folder.

cur or . (period)
Current message. This is the default.

first First message in a folder.

last Last message in a folder.

new New message that is created.

next Message following the current message.

prev Message preceding the current message

The default message is the current message in the current folder. When you specify several messages, the first message forwarded becomes the current message. When you specify a folder, that folder becomes the current folder.

-noannotate Prevents annotation of the original message. This flag is the default.

-nodraftfolder Places the draft in the *UserMhDirectory*/**draft** file.

-noedit Suppresses the initial edit.

-noformat Prevents reformatting of the messages being forwarded. This flag is the default.

-noinplace Prevents annotation in place. This flag is the default.

-nowhatnowproc Prevents interactive processing of the **forw** command. With this flag, no editing occurs.

-volume *Number* Specifies the volume number of the digest. The default volume number is the current value of the `DigestName-volume-list` entry in the *UserMhDirectory*/**context** file.

-whatnowproc *Program* Starts the specified program to guide you through the forwarding tasks.

Note: If you specify the **whatnow** command for *Program*, the **forw** command starts an internal **whatnow** procedure instead of a program with the file name **whatnow**.

Profile Entries

The following entries are entered in the *UserMhDirectory*/**.mh_profile** file:

`Current-Folder:` Sets the default current folder.

`Draft-Folder:` Sets the default folder for drafts.

`Editor:` Sets the default editor.

`fileproc:` Specifies the program used to refile messages.

`mhlproc:` Specifies the program used to filter messages being forwarded.

`Msg-Protect:` Sets the protection level for the new message files.

`Path:` Specifies the *UserMhDirectory*.

`whatnowproc:` Specifies the program used to prompt `What now?` questions.

Security

Attention RBAC users and Trusted AIX users: This command can perform privileged operations. Only privileged users can run privileged operations. For more information about authorizations and privileges, see Privileged Command Database in *Security*. For a list of privileges and the authorizations associated with this command, see the **/etc/security/privcmds** database file.

Examples

1. To forward the current message to another person, enter:

   ```
   forw
   ```

   The system prompts you to enter information in the header fields. To skip a field, press the Enter key. You must enter information in the `To:` field. The system responds with:

   ```
   ---------Enter initial text
   ```

   Enter the text you want displayed before the text of the forwarded message, and press the Ctrl-D key sequence. The text of the forwarded message is displayed, and you are prompted with `What now?` Enter `send` after the `What now?` prompt to forward the message.
2. To forward message `5` from the `inbox` folder, enter:

   ```
   forw    +inbox   5
   ```

Files

/etc/mh/digestcomps	Defines the MH default message form when the **-digest** flag is specified.
/etc/mh/mhl.forward	Contains the default MH message filter.
UserMhDirectory/**digestcomps**	Specifies a user's default message form when the **-digest** flag is specified. (If it exists, it overrides the MH default message filter.)
UserMhDirectory/**forwcomps**	Contains a user's default message form.
UserMhDirectory/**mhl.forward**	Contains a user's default message filter. (If it exists, it overrides the MH default message filter.)
/usr/bin/forw	Contains the executable form of the **forw** command.
$HOME/.mh_profile	Contains the file that customizes MH for an individual user.
UserMhDirectory/**draft**	Contains the draft created for editing messages.
/etc/mh/forwcomps	Defines components for the messages created by the **forw** command.

Related Information

The **anno** command, **comp** command, **dist** command, **mhl** command, **repl** command, **whatnow** command.

The **mh_alias** file format, **mh_profile** file format.

Mail applications in *Networks and communication management*.

---

fpm Command

Purpose

Manages the permissions on commands and daemons owned by privileged users with setuid or setgid permissions.

Syntax

fpm [**-l** *level* [**-f** *file*] [[**-c**] [**-p**]] [**-v**]] | [**-s**] | [**-q**] | [**-?**]

Description

The **fpm** command allows administrators to harden their system by disabling the setuid and setgid bits on many commands in the operating system. This command is intended to remove the setuid permissions from commands and daemons owned by privileged users, but you can also customize it to address the specific needs of unique computer environments.

The setuid programs on the base AIX operating system have been grouped to allow for levels of hardening. This grouping allows administrators to choose the level of hardening according to their system environment. Additionally, you can use the **fpm** command to customize the list of programs that need to be disabled in your environment. You must review the levels of disablement and choose the right level for your environment.

Changing execution permissions of commands and daemons with the **fpm** command affects non-privileged users, denying their access to these commands and daemons or functions of the commands and daemons. Additionally, other commands that call or depend on these commands and daemons can be affected. Any user-created scripts that depend on commands and daemons with permissions that were altered by the **fpm** command cannot operate as expected when run by non-privileged users. Give full consideration to the effect and potential impact of modifying default permissions of commands and daemons.

You must perform appropriate testing before using this command to change the execution permissions of commands and daemons in any critical computer environment. If you encounter problems in an environment where execution permissions have been modified, restore the default permissions and recreate the problem in this default environment to ensure the issue is not due to lack of appropriate execution permissions.

The **fpm** command provides the capability to restore the original AIX installation default permissions using the **-l default** flag.

Additionally, the **fpm** command logs the permission state of the files prior to changing them. The **fpm** log files are created in the **/var/security/fpm/log/***date_time* file. If necessary, you can use these log files to restore the system's file permissions recorded in a previously saved log file.

When the **fpm** command is used on files that have extended permissions, it disables the extended permissions, though any extended permission data that existed prior to the **fpm** invocation is retained in the extended ACL.

Customized configuration files can be created and enacted as part of the high, medium, low, and default settings. File lists can be specified in the **/usr/lib/security/fpm/custom/high/*** directory, the **/usr/lib/security/fpm/custom/medium/*** directory, and the **/usr/lib/security/fpm/custom/default/*** directory. To take advantage of this feature, create a file containing a list of files that you want to be automatically processed in addition to the **fpm** commands internal list. When the **fpm** command is run, it also processes the lists in the corresponding customized directories. To see an example of the format for a customized file, view the **/usr/lib/security/fpm/data/high_fpm_list** file. The default format can be viewed in the **/usr/lib/security/fpm/data/default_fpm_list.example** file. For the customization of the **-l low** flag, the **fpm** command reads the same files in the **/usr/lib/security/fpm/custom/medium** directory, but only removes the setgid permissions, whereas the **-l medium** flag removes both the setuid and setgid permissions.

The **fpm** command cannot run on TCB-enabled hosts.

Flags

-l *level*
Specifies that the file permissions are changed according to the level specified.

-l high High-level security. This flag removes the setuid and setgid permissions for computer systems that fall into the category of high-level security. This flag uses the list of files in the **/usr/lib/security/fpm/data/high_fpm_list** file and the **/usr/lib/security/fpm/custom/high/*.*** file as input by default, but an alternate input file can be selected with the **-f** flag.

-l medium
Medium-level security. This flag removes the setuid and setgid permissions for computer systems that fall into the category of medium-level security. This flag uses the list of files in the **/usr/lib/security/fpm/data/med_fpm_list** file and the **/usr/lib/security/fpm/custom/med/*.*** file as input by default. An alternate input file can be selected with the **-f** flag.

-l low Low-level security. This flag removes only the setuid permission for computer systems that fall into the category of low-level security. This flag uses the list of files in the **/usr/lib/security/fpm/data/med_fpm_list** file and the **/usr/lib/security/fpm/custom/med/*.*** file as input by default. An alternate input file can be selected with the **-f** flag.

-l default
Returns the system commands previously modified by the **fpm** command to their default out-of-the-box permissions, if the commands were previously altered using the level of high, medium or low. This option reads the **/usr/lib/security/fpm/custom/default/*.*** file and sets the permissions defined in the file.

-s
Displays the status of the changes last made by the **fpm** command. The status is written in the **/usr/lib/security/fpm/data/status_fpm** file. The security level is represented as a whole integer from 1-5 (inclusive).

-f *file*
Allows the specification of a file list to override the default input file, where the *file* parameter is a file name containing the list of files to be used as input. This flag must be used along with the **-l high|medium|low|default** or the **-c** flag. When using a level of high, medium or low, the input file format is as follows:

full_path/*filename*

For example, `/usr/sbin/foo`.

When used with the **-l default** flag, the input file format is as follows:

octet_permissions full_path/*filename*

There must be a space between the *octet_permissions* variable and the *full_path* variable. For example, `0750 /usr/sbin/foo`.

The **-f** format allows for the specific control of the list of files being affected.

-c
Checks the files permissions, but takes no action. The **fpm** command returns 0 if no files were found out of compliance. If one or more files contain non-compliant permissions, this option lists the non-compliant file(s) and returns 1. This flag must be used with the **-l** *level* option. For example, if the **-c** and the **-l high** flags are used together, the **fpm** command checks the files listed in the **/usr/lib/security/fpm/data/high_fpm_list** file and removes their setuid and setgid permissions. The **-f** *file* flag can also be used with the **-c** option.

-v
Verbose output.

-p
Previews the changes the **fpm** command is to make, but takes no action. This flag must be used in conjunction with the **-l** *level* flag.

-q
Quit mode, which minimizes output and suppresses warnings.

-?
Prints the usage statement.

Exit Status

0	Success.
Non-zero	Failure or partial failure. Use the **-v** flag for more details.

Security

The **fpm** command reduces the number of commands with setuid and setgid permissions.

Examples

1. To apply the **fpm** command's low level security settings, enter:
   ```
   fpm –l low
   ```
   This command also processes any file list in the **/usr/lib/security/fpm/custom/med/** directory.
2. To check if the system commands are presently set to **fpm** low-level permissions, enter:
   ```
   fpm –c –l low
   ```
   This command reports any file with permissions out of conformance.
3. To restore the traditional out-of-the-box default permissions, enter:
   ```
   fpm –l default
   ```
   This command also processes any file list in the **/usr/lib/security/fpm/custom/default/** directory.
4. To list, or give a preview of what permission changes are to be done to make the system compliant with the **fpm** command's high-level security without changing any file permissions, enter:
   ```
   fpm -l high –p
   ```
   This command also previews any file list in the **/usr/lib/security/fpm/custom/high/** directory.
5. To apply the **fpm** command's high level security settings, enter:
   ```
   fpm –l high
   ```
   This command also processes any file list in the **/usr/lib/security/fpm/custom/high/** directory.
6. To list the current status of the system as changed through the **fpm** command, enter:
   ```
   fpm –s
   ```
7. If the **fpm -l** *level* command was run on 7 January 2007 at 8:00 a.m., then the permission state of the affected files was captured by the **fpm** command before it made any changes. To restore the file permissions to their state of 7 January 2007 at 8:00 a.m., enter:
   ```
   fpm –l default –f /var/security/fpm/log/01072007_08:00:00
   ```

Files

/usr/lib/security/fpm/data/default_list_example	Contains the default out-of-the-box permissions and files.
/usr/lib/security/fpm/data/high_fpm_list	Contains the list of files whose permissions can be changed with the **-l high** flag.
/usr/lib/security/fpm/data/med_fpm_list	Contains the list of files whose permissions can be changed with the **-l medium** or **-l low** flag.
/usr/lib/security/fpm/custom/high/*	Files in this directory can be used as user-configured input when the **-l high** level is selected. These files must contain a list of files, from which the **fpm** command removes setuid and setgid permissions.
/usr/lib/security/fpm/custom/medium/*	Files in this directory serve the same function as the high-level directory, but are used with the **-l medium** flag and the **-l low** flag.

/usr/lib/security/fpm/custom/default/*	Files in this directory serve the same function as the high-level directory, but are used with the **-l default** flag. **Note:** These files must be in the same format as the **/usr/lib/security/fpm/data/default_list_example** file.
/usr/lib/security/fpm/data/status_fpm	Contains the status of the file permissions changed from the last run of the **fpm** command.
/var/security/fpm/log/*date_time*	Contains the list of files changed by the **fpm** command corresponding to the data and the time at which the command was run. This file can be used as the input file of the **-f** flag to restore permissions to this instance.

Related Information

The **aixpert** command.

The AIX Security Expert in *Security*.

---

frcactrl Command

Purpose

Controls and configures FRCA.

Syntax

frcactrl { **load** | **unload** }
frcactrl open *Ip_Address Port* [*Virtual_Host*] *Server_Name Virtual_Root Log_File*
frcactrl close *Ip_Address Port* [*Virtual_Host*]
frcactrl loadfile *Ip_Address Port* [*Virtual_Host*] *Document_Root File ...*
frcactrl stats [reset] [*Interval*]
frcactrl logging *Ip_Address Port* [*Virtual_Host*] { on | off } [*Format*] [*CPU_Id*]
frcactrl { **start** | **stop** } *Ip_Address Port* [*Virtual_Host*]
frcactrl revaltimeout *Ip_Address Port* [*Virtual_Host*] [*Seconds*]
frcactrl pctonintr [*Percentage*]
frcactrl set { option=*value* }
frcactrl get
frcactrl default [*option*]

Description

The **frcactrl** command controls and configures the FRCA kernel extension. The kernel extension must be loaded before starting any Web servers that want to use FRCA.

Subcommands

load Loads the FRCA kernel extension if not loaded.

unload
Unloads the FRCA kernel extension if loaded.

open *Ip_Address Port* **[** *Virtual_Host* **]** *Server_Name Virtual_Root Log_File*
Opens and configures an FRCA instance under the name *Server_Name* for IP address *Ip_Address* on port *Port*. The *Virtual_Root* parameter specifies the directory where the Web data starts. The requests will be logged in the file specified by *Log_File*. This filename must be fully qualified.

Note: FRCA only supports one log file. When running more than one Web server on a system with FRCA, all requests will be logged to the same file.

close *Ip_Address Port* **[** *Virtual_Host* **]**
Closes the FRCA instance associated with the specified IP address and port.

loadfile *Ip_Address Port* **[** *Virtual_Host* **]** *Document_Root File* **...**
Loads the specified file(s) into the FRCA / Network Buffer Cache. The IP and Port number at which the FRCA instance has been opened earlier must be specified here along with the document root and the file(s) to be loaded.

stats [reset] [*Interval* **]**
Displays FRCA statistics. The optional **reset** subcommand clears (zeros) the statistics. You can display the statistics at a regular interval by specifying the duration of the interval in seconds with the *Interval* parameter.

logging *Ip_Address Port* **[** *Virtual_Host* **] { on | off } [** *Format* **] [** *CPU_Id* **]**
Turns logging of request served by an FRCA instance bound to the specified *Ip_Address* and *Port* on or off. The format can be one of CLF, V-CLF, or ECLF (Common Log Format, Virtual Host & CLF, Extended CLF). The FRCA logging thread can also be bound to a particular CPU by specifying the optional *CPU_Id* parameter on multiprocessor machines.

start *Ip_Address Port* **[** *Virtual_Host* **]**
Enables the kernel get engine to serve requests sent to the specified IP and port.

stop *Ip_Address Port* **[** *Virtual_Host* **]**
Disables the kernel get engine for the specified IP and port.

revaltimeout *Ip_Address Port* **[** *Virtual_Host* **] [** *Seconds* **]**
Changes the revalidation timeout value for an FRCA instance at the specified address and port. The timeout value must be specified in seconds.

pctonintr [*Percentage* **]**
Controls the percentage of CPU time that can be spent in interrupt context. If this value is too low then FRCA will send requests up to Web server more often since it always executes in interrupt context. Any value >= 100 will result in FRCA serving every request that is cached in the FRCA cache.

set {option=*value***}**
Sets the specified FRCA option to the value. The only option currently available is **frca_hashsz** which sets the number of slots in the FRCA hash table to the specified value. The default value of **frca_hashsz** is 12841. If changed, the value used must be prime as this results in a more even distribution of hash table entries.

get Displays all FRCA options available along with their current values. Only one option called **frca_hashsz** currently exists.

default [*option***]**
Sets the value of all options to their default values when used without specifying an option name. If an option name is specified it sets only the value of the specified option to its default.

Security

Attention RBAC users and Trusted AIX users: This command can perform privileged operations. Only privileged users can run privileged operations. For more information about authorizations and privileges, see Privileged Command Database in *Security*. For a list of privileges and the authorizations associated with this command, see the **/etc/security/privcmds** database file.

Examples

1. The following are examples of using the **open** subcommand:

```
frcactrl open 9.1.1.1 80 ici imgcache01 /htdocs /logs/frca.log bin
frcactrl open 9.1.1.2 80 ici imgcache02 /htdocs /logs/frca.log bin
```

In the above examples ″ici″ is the virtual host name which could be used to access one of the mirrors imgcache01 or imgcache02. The IP address may be 0.0.0.0 if the Web server is not bound to a specific IP address.

2. To close the FRCA instance associated with IP address 9.1.1.1 and port 80, type:
   ```
   frcactrl close 9.1.1.1 80
   ```
3. To load the content of files /a/b/c/d and /a/b/c/e with URLs /d and /e, type:
   ```
   frcactrl loadfile /a/b/c /a/b/c/d e
   ```
4. To display the FRCA statistics, type:
   ```
   frcactrl stats
   ```

   This will cause the FRCA statistics to be displayed. They will look similar to this:
   ```
          Total    Deferred       Cache       Cache    Resource
      Requests    Requests        Hits      Misses      Errors
   ------------------------------------------------------------
    1024065396         227  1024065168           1           0
   ```
5. This examples shows how to use the **start** subcommand for virtual host ″ici″:
   ```
   frcactrl start 9.1.1.1 80 ici
   ```

   Note: The virtual host parameter is optional.
6. To disable the kernel get engine for port 80 on IP address 9.1.1.1 on virtual host ″ici″, type:
   ```
   frcactrl stop 9.1.1.1 80 ici
   ```
7. The following example sets the revalidation timeout value for the FRCA instance at port 80 of IP address 9.1.1.1 to 100 seconds:
   ```
   frcactrl revaltimeout 9.1.1.1 80 100
   ```
8. To allow the CPU to spend 98 percent of its time in interrupt context, type:
   ```
   frcactrl pctonintr 98
   ```
9. To set the value of the **frca_hashsz** option to 24499, type:
   ```
   frcactrl set frca_hashsz=24499
   ```
10. To set the value of **frca_hashsz** to its default, type:
    ```
    frcactrl default frca_hashsz
    ```

Files

/usr/bin/frcactrl

---

from Command

Purpose

To determine whom mail is from.

Syntax

from [**-d** *Directory*] [**-s** *Sender*] [*user*]

Description

The **from** command displays the message headings in your mailbox file to show you whom mail is from. If you specify *user*, the *user* mailbox is examined instead of your own (provided that you have read permission to user's mailbox).

Flags

-d *Directory* Specifies the system mailbox directory.

-s *Sender* Prints message headers only for mail sent by *Sender*.

Parameters

user Specifies the *user* mailbox that is examined instead of your own (provided that you have read permission to the user's mailbox).

Security

Attention RBAC users and Trusted AIX users: This command can perform privileged operations. Only privileged users can run privileged operations. For more information about authorizations and privileges, see Privileged Command Database in *Security*. For a list of privileges and the authorizations associated with this command, see the **/etc/security/privcmds** database file.

Examples

1. To display the message headings in your mailbox, enter:
   ```
   from
   ```
   The names of the senders and message dates are displayed.
2. To display the message headings for mail sent by a specific user, enter:
   ```
   from -s dale
   ```
   In this example, only the message headings of the messages sent from user `dale` are displayed.
3. To display the message headings in a specific user's mailbox, enter:
   ```
   from   dawn
   ```
   In this example, the message headings from user `dawn's` mailbox are displayed (provided that you have read permission to `dawn's` mailbox).
4. To view all messages `bob` received from `jane`, enter:
   ```
   from -d /var/spool/mail/bob -s jane root;
   ```
   This allows you to see all messages that `bob` received from `jane`, provided you have the permissions (such as root).

Files

/var/spool/mail/*	System mailboxes for all users.
/usr/bin/from	User mailbox files.

Related Information

The **mail** command.

Mail applications in *Networks and communication management*.

---

fsck Command

Purpose

Checks file system consistency and interactively repairs the file system.

Syntax

fsck [**-n**] [**-p**] [**-y**] [**-d***BlockNumber*] [**-f**] [**-i***i-NodeNumber*] [**-o** *Options*] [**-t***File*] [**-V** *VfsName*] [*FileSystem1 - FileSystem2 ...*]

Description

Attention: Always run the **fsck** command on file systems after a system malfunction. Corrective actions may result in some loss of data. The default action for each consistency correction is to wait for the operator to enter `yes` or `no`. If you do not have write permission for an affected file system, the **fsck** command defaults to a `no` response in spite of your actual response.

Notes:

1. The **fsck** command does not make corrections to a mounted file system.
2. The **fsck** command can be run on a mounted file system for reasons other than repairs. However, inaccurate error messages may be returned when the file system is mounted.

The **fsck** command checks and interactively repairs inconsistent file systems. You should run this command before mounting any file system. You must be able to read the device file on which the file system resides (for example, the **/dev/hd0** device). Normally, the file system is consistent, and the **fsck** command merely reports on the number of files, used blocks, and free blocks in the file system. If the file system is inconsistent, the **fsck** command displays information about the inconsistencies found and prompts you for permission to repair them.

The **fsck** command is conservative in its repair efforts and tries to avoid actions that might result in the loss of valid data. In certain cases, however, the **fsck** command recommends the destruction of a damaged file. If you do not allow the **fsck** command to perform the necessary repairs, an inconsistent file system may result. Mounting an inconsistent file system may result in a system crash.

If a JFS2 file system has snapshots, the **fsck** command will attempt to preserve them. If this action fails, the snapshots cannot be guaranteed to contain all of the before-images from the snapped file system. The **fsck** command will delete the snapshots and the snapshot logical volumes. Internal snapshots are deleted if the **fsck** command modifies the file system.

If you do not specify a file system with the *FileSystem* parameter, the **fsck** command checks all file systems listed in the **/etc/filesystems** file for which the **check** attribute is set to True. You can enable this type of checking by adding a line in the stanza, as follows:

```
check=true
```

You can also perform checks on multiple file systems by grouping the file systems in the **/etc/filesystems** file. To do so, change the check attribute in the **/etc/filesystems** file as follows:

```
check=Number
```

The *Number* parameter tells the **fsck** command which group contains a particular file system. File systems that use a common log device should be placed in the same group. File systems are checked, one at a time, in group order, and then in the order that they are listed in the **/etc/filesystems** file. All `check=true` file systems are in group `1`. The **fsck** command attempts to check the root file system before any other file system regardless of the order specified on the command line or in the **/etc/filesystems** file.

The **fsck** command checks for the following inconsistencies:

- Blocks or fragments allocated to multiple files.
- i-nodes containing block or fragment numbers that overlap.
- i-nodes containing block or fragment numbers out of range.
- Discrepancies between the number of directory references to a file and the link count of the file.
- Illegally allocated blocks or fragments.

- i-nodes containing block or fragment numbers that are marked free in the disk map.
- i-nodes containing corrupt block or fragment numbers.
- A fragment that is not the last disk address in an i-node. This check does not apply to compressed file systems.
- Files larger than 32KB containing a fragment. This check does not apply to compressed file systems.
- Size checks:
  - Incorrect number of blocks.
  - Directory size not a multiple of 512 bytes.

  These checks do not apply to compressed file systems.
- Directory checks:
  - Directory entry containing an i-node number marked free in the i-node map.
  - i-node number out of range.
  - Dot (.) link missing or not pointing to itself.
  - Dot dot (..) link missing or not pointing to the parent directory.
  - Files that are not referenced or directories that are not reachable.
- Inconsistent disk map.
- Inconsistent i-node map.

Orphaned files and directories (those that cannot be reached) are, if you allow it, reconnected by placing them in the **lost+found** subdirectory in the root directory of the file system. The name assigned is the i-node number. If you do not allow the **fsck** command to reattach an orphaned file, it requests permission to destroy the file.

In addition to its messages, the **fsck** command records the outcome of its checks and repairs through its exit value. This exit value can be any sum of the following conditions:

0 All checked file systems are now okay.
2 The **fsck** command was interrupted before it could complete checks or repairs.
4 The **fsck** command changed the file system; the user must restart the system immediately.
8 The file system contains unrepaired damage.

When the system is booted from a disk, the boot process explicitly runs the **fsck** command, specified with the **-f** and **-p** flags on the **/**, **/usr**, **/var**, and **/tmp** file systems. If the **fsck** command is unsuccessful on any of these file systems, the system does not boot. Booting from removable media and performing maintenance work will then be required before such a system will boot.

If the **fsck** command successfully runs on **/**, **/usr**, **/var**, and **/tmp**, normal system initialization continues. During normal system initialization, the **fsck** command specified with the **-f** and **-p** flags runs from the **/etc/rc** file. This command sequence checks all file systems in which the **check** attribute is set to True (`check=true`). If the **fsck** command executed from the **/etc/rc** file is unable to guarantee the consistency of any file system, system initialization continues. However, the mount of any inconsistent file systems may fail. A mount failure may cause incomplete system initialization.

Note: By default, the **/**, **/usr**, **/var**, and **/tmp** file systems have the **check** attribute set to False (`check=false`) in their **/etc/filesystem** stanzas. The attribute is set to False for the following reasons:

1. The boot process explicitly runs the **fsck** command on the **/**, **/usr**, **/var**, and **/tmp** file systems.
2. The **/**, **/usr**, **/var**, and **/tmp** file systems are mounted when the **/etc/rc** file is executed. The **fsck** command will not modify a mounted file system. Furthermore, the **fsck** command run on a mounted file system produces unreliable results.

You can use the File Systems application in Web-based System Manager (wsm) to change file system characteristics. You could also use the System Management Interface Tool (SMIT) **smit fsck** fast path to run this command.

Flags

-d*BlockNumber* Searches for references to a specified disk block. Whenever the **fsck** command encounters a file that contains a specified block, it displays the i-node number and all path names that refer to it. For JFS2 filesystems, the i-node numbers referencing the specified block will be displayed but not their path names.″

-f Performs a fast check. Under normal circumstances, the only file systems likely to be affected by halting the system without shutting down properly are those that are mounted when the system stops. The **-f** flag prompts the **fsck** command not to check file systems that were unmounted successfully. The **fsck** command determines this by inspecting the **s_fmod** flag in the file system superblock.

This flag is set whenever a file system is mounted and cleared when it is unmounted successfully. If a file system is unmounted successfully, it is unlikely to have any problems. Because most file systems are unmounted successfully, not checking those file systems can reduce the checking time.

-i*i-NodeNumber* Searches for references to a specified i-node. Whenever the **fsck** command encounters a directory reference to a specified i-node, it displays the full path name of the reference.

-n Assumes a `no` response to all questions asked by the **fsck** command; does not open the specified file system for writing.

-o *Options* Passes comma-separated options to the **fsck** command. The following options are currently supported for JFS (these options are obsolete for newer file systems and can be ignored):

mountable
Causes the **fsck** command to exit with success, returning a value of 0, if the file system in question is mountable (clean). If the file system is not mountable, the **fsck** command exits returning with a value of 8.

mytype
Causes the **fsck** command to exit with success (0) if the file system in question is of the same type as either specified in the **/etc/filesystems** file or by the **-V** flag on the command line. Otherwise, 8 is returned. For example, `fsck -o mytype -V jfs /` exits with a value of 0 if **/** (the root file system) is a journaled file system.

-p Does not display messages about minor problems but fixes them automatically. This flag does not grant the wholesale license that the **-y** flag does and is useful for performing automatic checks when the system is started normally. You should use this flag as part of the system startup procedures, whenever the system is being run automatically. If the primary superblock is corrupt, the secondary superblock is verified and copied to the primary superblock.

-t*File* Specifies a *File* parameter as a scratch file on a file system other than the one being checked, if the **fsck** command cannot obtain enough memory to keep its tables. If you do not specify the **-t** flag and the **fsck** command needs a scratch file, it prompts you for the name of the scratch file. However, if you have specified the **-p** flag, the **fsck** command is unsuccessful. If the scratch file is not a special file, it is removed when the **fsck** command ends.

-V *VfsName* Uses the description of the virtual file system specified by the *VFSName* variable for the file system instead of using the **/etc/filesystems** file to determine the description. If the **-V** *VfsName* flag is not specified on the command line, the **/etc/filesystems** file is checked and the **vfs=***Attribute* of the matching stanza is assumed to be the correct file system type.

-y Assumes a yes response to all questions asked by the **fsck** command. This flag lets the **fsck** command take any action it considers necessary. Use this flag only on severely damaged file systems.

Security

Attention RBAC users and Trusted AIX users: This command can perform privileged operations. Only privileged users can run privileged operations. For more information about authorizations and privileges, see Privileged Command Database in *Security*. For a list of privileges and the authorizations associated with this command, see the **/etc/security/privcmds** database file.

Examples

1. To check all the default file systems, enter:

   ```
   fsck
   ```

   This command checks all the file systems marked `check=true` in the **/etc/filesystems** file. This form of the **fsck** command asks you for permission before making any changes to a file system.
2. To fix minor problems with the default file systems automatically, enter:

   ```
   fsck -p
   ```
3. To check a specific file system, enter:

   ```
   fsck /dev/hd1
   ```

   This command checks the unmounted file system located on the **/dev/hd1** device.

Files

/usr/sbin/fsck	Contains the **fsck** command.
/etc/filesystems	Lists the known file systems and defines their characteristics.
/etc/vfs	Contains descriptions of virtual file system types.
/etc/rc	Contains commands (including the **fsck** command) that are run when the system is started.

Related Information

The **dfsck** command, **fsdb** command, **istat** command, **mkfs** command, **ncheck** command, **rc** command, **shutdown** command.

The **filesystems** file, **filsys.h** file.

The File systems in *Operating system and device management* explains file system types, management, structure, and maintenance.

For information on installing the Web-based System Manager, see Chapter 2: Installation and System Requirements in *AIX Version 6.1 Web-based System Manager Administration Guide*.

The System management interface tool in *Operating system and device management* explains the SMIT structure, main menus, and tasks.

---

fsck_cachefs Command

Purpose

Checks the integrity of data cached with CacheFS.

Syntax

fsck_cachefs [**-m**] [**-o noclean**] *cache_directory*

Description

The CacheFS version of the **fsck** command checks the integrity of a cache directory. By default it corrects any CacheFS problems it finds. There is no interactive mode. The most likely invocation of **fsck_cachefs** for CacheFS filesystems is at boot time from an entry in **/etc/rc.nfs**.

Flags

-m Check, but do not repair.
-o noclean Force a check on the cache even if there is no reason to suspect there is a problem.

Examples

To force a check on the cache directory, enter:

```
fsck_cachefs -o noclean /cache3
```

---

fsdb Command

Purpose

Debugs file systems.

Syntax

fsdb *FileSystem* [**-**]

Description

The **fsdb** command enables you to examine, alter, and debug a file system, specified by the *FileSystem* parameter. The command provides access to file system objects, such as blocks, i-nodes, or directories. You can use the **fsdb** command to examine and patch damaged file systems. Key components of a file system can be referenced symbolically. This feature simplifies the procedures for correcting control-block entries and for descending the file system tree.

To examine a file system, specify it by a block device name, a raw device name, or a mounted file system name. In the last case, the **fsdb** command determines the associated file system name by reading the **/etc/filesystems** file. Mounted file systems cannot be modified.

The **fsdb** command has a different interface for a JFS file system and a JFS2 file system. The following explains how to use **fsdb** with a JFS file system. See JFS2 Subcommands for information about JFS2 subcommands.

If the file system specified is a JFS2 snapshot, the **fsdb** command enables examination and modification of the snapshot superblock, snapshot map, block map xtree copy, and segment headers. See JFS2 Snapshot Subcommands for information about JFS2 snapshot subcommands.

The subcommands for the **fsbd** command allow you to access, view, or change the information in a file system. Any number you enter in the subcommand is considered decimal by default, unless you prefix it with either 0 to indicate an octal number or 0x to indicate a hexadecimal number. All addresses are printed in hexadecimal.

Because the **fsdb** command reads and writes one block at a time, it works with raw as well as with block I/O.

Flag

- Disables the error checking routines used to verify i-nodes and block addresses. The **O** subcommand switches these routines on and off. When these routines are running, the **fsdb** command reads critical file system data from the superblock. The obtained information allows the **fsdb** command to access the various file system objects successfully and to perform various error checks.

Subcommands

The **fsdb** subcommands are requests to locate and display or modify information in the file system. The main categories of subcommands are:

Category	Function
Location	Access the information in the file system.
Display	View the information in the file system.
Modification	Change the information in the file system.

In addition, there are a few miscellaneous subcommands.

Location Subcommands

There are two types of location subcommands:

Number[**I** | **M** | **i** | **b**]

OR

d*DirectorySlot*

The first type consists of a number, optionally followed by an address specification. The address specification defines how the preceding number is to be interpreted. There are four address specifications corresponding to four different interpretations of the *Number* variable:

I I-node map block number
M Disk map block number
i I-node number
b Fragment number

Depending on the address specification (or absence of it), this type of location subcommand accesses information as follows:

Number	Accesses data at the absolute byte offset specified by the *Number* variable.
*MapBlockNumber***I**	Accesses the i-node map block indicated by the *MapBlockNumber* variable.
*MapBlockNumber***M**	Accesses the disk map block indicated by the *MapBlockNumber* variable.
*InodeNumber***i**	Accesses the i-node indicated by the *InodeNumber* variable.
*FragmentNumber***b**	Accesses the file system block indicated by the *FragmentNumber* variable. A fragment number consists of a block address and an encoded length. A complete fragment address is 32 bits in length. The low-order 28 bits are the beginning fragment address. The fragment length is encoded in the remaining 4 bits; it is encoded as the number of fragments less than a full block. For example, on a file system consisting of 1024-byte fragments, the address `0x2000010f` references a block that begins at 1KB block number `0x10f` and is 2KB in length. In contrast, on a file system of 512-byte fragments, the address `0x2000010f` references a block that begins at 512-byte block `0x10f` and is 3072 (512 * 6) bytes in length.

The second type of location subcommand is used to access directory entries. The subcommand consists of the character **d** followed by a directory-slot number. Directory-slot numbers start at 0 for each block of the associated i-node.

This type of location subcommand accesses information as follows:

d*DirectorySlot* Accesses the directory entry indexed by the *DirectorySlot* variable for the current i-node. Only allocated directory entries can be manipulated using this location subcommand.

Display Subcommands

To view information relative to the address specification, use a display subcommand comprised of one of the display facilities in conjunction with one of the display formats, as follows:

p[*Number*]{ **i** | **d** | **o** | **e** | **c** | **b** | **y** | **M** | **I** | **x** | **s** | **D** }

OR

f[*Number*]{ **i** | **d** | **o** | **e** | **c** | **b** | **y** | **M** | **I** | **x** | **s** | **D** }

The display facilities are:

p Indicates a general facility. Use the general display subcommand to display data relative to the current address. If you enter a number after the **p** symbol, the **fsdb** command displays that number of entries. A check is made to detect block boundary overflows. If you enter 0 or * (asterisk), the **fsdb** command displays all entries to the end of the current fragment.

f Indicates a file facility. Use the file display subcommand to display data blocks associated with the current i-node. If you enter a number after the **f** symbol, the **fsdb** command displays that block of the file. Block numbering begins at 0. The display format follows the block number. If you enter **f** without a block number, the **fsdb** command defaults to displaying block 0 of the current i-node.

The display formats for either facility are:

i Displays as i-nodes.
d Displays as directories.
o Displays as octal words.
e Displays as decimal words.
c Displays as characters.
b Displays as octal bytes.
y Displays as hexadecimal bytes.
M Displays as disk map entries.
I Displays as i-node map entries.
x Displays as hexadecimal words.
S Displays as single indirect blocks.
D Displays as double indirect blocks.

The chosen display facility and display format remain in effect during the processing of the **fsdb** command until explicitly changed. You may receive an error message indicating improper alignment if the address you specify does not fall on an appropriate boundary.

If you use the *Number*, *MapBlockNumber***I**, or *FragmentNumber***b** location subcommands to access i-node information, you can step through the data, examining each byte, word, or double word. Select the desired display mode by entering one of the following subcommands:

B Begins displaying in byte mode.
D Begins displaying in double-word mode.
W Begins displaying in word mode.

You can move forward or backward through the information. The boundary advances with the display screen and is left at the address of the last item displayed. The output can be ended at any time by pressing the INTERRUPT key. The following symbols allow movement through the information:

+ *Number* Moves forward the specified number of units currently in effect.
-*Number* Moves backward the specified number of units currently in effect.

The following symbols allow you to store the current address and return to it conveniently:

> Stores the current address.
< Returns to the previously stored address.

You can use dots, tabs, and spaces as subcommand delimiters, but they are only necessary to delimit a hexadecimal number from a subcommand that could be interpreted as a hexadecimal digit. Pressing the Enter key (entering a blank line) increments the current address by the size of the data type last displayed. That is, the address is set to the next byte, word, double word, directory entry, or i-node, allowing you to step through a region of a file system.

The **fsdb** command displays information in a format appropriate to the data type. Bytes, words, and double words are displayed as a hexadecimal address followed by the hexadecimal representation of the data at that address and the decimal equivalent enclosed in parentheses. The **fsdb** command adds a **.B** or **.D** suffix to the end of the address to indicate a display of byte or double word values. It displays directories as a directory slot offset followed by the decimal i-node number and the character representation of the entry name. It displays i-nodes with labeled fields describing each element. The environment variables control the formats of the date and time fields.

Modification Subcommands

You can modify information relative to the address specification by using a field specification (for fields in the i-node and fields in the directory). The general form for assigning new values is: *mnemonic operator new-value*, where the *mnemonic* parameter represents one of the fields described in the following list:

The following mnemonics are used for the names of the fields of an i-node and refer to the current working i-node:

`md` Permission mode
`ln` Link count
`uid` User number
`gid` Group number
`sz` File size
`a`*Number* Data block numbers (0 to 8) where the *Number* parameter can be a location subcommand
`at` Access time
`mt` Modification time
`maj` Major device number
`min` Minor device number

The following mnemonics refer to the i-node and disk maps:

`mf` Map free count
`ms` Map size
`mp` Permanent allocation bit map
`mw` Working allocation bit map

The following mnemonics are used for the names of the fields in directories:

`rl` Length of directory entry record
`nl` Length of directory name
`nm` Directory name

Valid values of the *Operator* parameter include:

Note: A file system must be unmounted before attempting to modify it.

= Assigns the *New-Value* parameter to the specified *Mnemonic* parameter.
=+ Increment the *Mnemonic* parameter by the specified *New-Value* parameter. The default *New-Value* parameter is a value of one.
=- Decrease the *Mnemonic* by the specified *New-Value*. The default *New-Value* is a value of one.
=″ Assigns the character string specified by the *New-Value* parameter to the specified *Mnemonic* parameter. If the current display format is the **d** address specification for directory and a mnemonic is not specified, the directory name is changed. The new directory name cannot be longer than the previous directory name.

Miscellaneous Subcommands

Miscellaneous subcommands are:

q Quits.

x*n* Expands a directory by *n* bytes where *n* plus the current size of the directory is not greater than the current directory's fragment in bytes.
! Escapes to the shell.
O Toggles error checking.

JFS2 Subcommands

These subcommands can be entered by their entire name or by using a subset of the name. At least the bold letters must be entered.

a[lter] <block> <offset> <hex string>	Alters disk data.
b[map] [<block number>]	Displays block allocation map.
dir[ectory] <inode number> [<fileset>] [R]	Displays directory entries.
d[isplay] [<block> [<offset> [<format> [<count>]]]]	Displays data.
dt[ree] {<block number> I<inode number>{a I f } }	Displays dtree nodes.
h[elp] [<command>]	Provides help on subcommands.
ia[g] [<IAG number>] [a I <fileset>]	Displays IAG pages.
im[ap] [a I <fileset>]	Displays inode allocation map.
i[node] [<inode number>] [a I <fileset>]	Displays inodes.
q[uit]	Exits fsdb.
su[perblock] [p I s]	Displays superblock.
x[tree] {<block number> I <inode number>{a I f}}	Displays xtree nodes.

a[lter] <block> <offset> <hex string>
where:

<block>	block number (decimal)
<offset>	offset within block (hex)
<hex string>	string of hex digits

Alters disk data. <hex string> should contain an even number of digits.

b[map] [<block numbers>]

Display Block Allocation Map.

<block number> Display the **dmap** page which describes this block number

Subcommands:

m	modify current node
u	visit upper level **bmap** page
l	visit left sibling
r	visit right sibling
w	display wmap
p	display pmap
s	display stree
x	exit subcommand mode

dir[ectory] <inode number> [<fileset>][R]

<inode number>	inode number of directory (decimal)
<fileset>	number, currently must be zero
R	recursively lists all subdirectories

Displays directory entries.

d[isplay] [<block> [<offset> [<format>[<count>]]]]

<block>	block number (decimal)
<offset>	offset within block (hex)
<format>	format in which to display data (see below)
<count>	number of objects to display (decimal)

Displays data in a variety of formats.

Format may be one of the following:

a	ascii	
i	inode	struct dinode
l	inode allocation map	iag_t
s	superblock	struct superblock
x	hexadecimal	

dt[ree] {<block number> | <inode number>{a | f}}

<block number>	block number containing a dtree page
<inode number>	inode number of directory (decimal)
{a | f}	'a' indicates inode number is an aggregate inode. 'f' indicates inode number is a fileset inode.

Displays root of the directory btree and enters a subcommand mode in which to navigate the btree.

Subcommands:

m	Modifies current node
f	Walks freelist entries
s	Displays specified slot entry
[0-9]+	Displays specified stbl entry
t	Displays formatted stbl
u	Visits parent node (not parent directory)
d	Visits child node
x	Exits subcommand mode

h[elp] [<command>]

<command>	command name

Prints help text. Lists all commands if no parameter.

ia[g] [<IAG number>] [a | <fileset>]

<IAG number>	IAG number (decimal)
a	use aggregate inode table
<fileset>	fileset number (currently must be zero)

Displays iag information and enters subcommand mode.

Subcommands:

e	Displays/modifies inode extents map
m	Modifies iag
p	Displays/modifies persistent map
w	Displays/modifies working map

im[ap] [a | <fileset>]

a	use aggregate inode table
<fileset>	fileset number (currently must be zero)

Display specified inode map and enters subcommand mode.

Subcommands:

e	Displays/modifies inode extents map
m	Modifies iag
p	Displays/modifies persistent map

i[node] [<inode number>] [a | <fileset>]

<inode number>	Inode number (decimal)
a	Use aggregate inode table
fileset	Fileset number (currently must be zero)

Displays inode information and enters subcommand mode.

Subcommands:

m	Modifies inode
t	Displays/modifies inode's b-tree
e	display/modify inode's EAs

Note: The **fsdb** command understands both the **v1** and the **v2** extended attribute formats. The behavior when viewing EAs is dependent on the format for the inode being viewed.

For **v1**, after displaying the inode's EAs you can modify its `pxdTable` or `eaDirectory` entries. Specify modify option and then the `pxdTable` or `eaDirectory` indicator and the offset into the table.

For **v2** the EAs are displayed using the **dtree** subcommand format. All of the **dtree** subcommands are then available for further action on the EAs.

q[uit] Exits fsdb.

su[perblock] [p | s]

p	Displays primary superblock
s	Displays secondary superblock

Displays superblock data.

x[tree] {<block number> | <inode number>{a | f} }

<block number>	block number (decimal)
<inode number>	inode number
{a | f}	'a' indicates inode number is an aggregate inode. 'f' indicates inode number is a fileset inode.

Displays one node of a xtree and enters a subcommand mode in which to navigate the xtree.

Subcommands:

m	Modifies current node
u	Visits parent node
d	Visits child node
n	Visits right sibling
p	Visits left sibling
s	Selects xad entry to view
x	Exits subcommand mode

JFS2 Snapshot Subcommands

These subcommands can be entered by their entire name or by using a subset of the name. At least the bold letters must be entered.

a[lter] <block> <offset> <hex string>	Alters disk data.
b[map]	Displays block map xtree copy.
d[isplay] [<block> [<offset> [<format> [<count>]]]]	Displays data.
h[elp] [<command>]	Provides help on subcommands.
q[uit]	Exits fsdb.
st[able] [<block number>]	Displays summary snapshot table.
s[map] <block number>	Displays snapshot bit map.
su[perblock]	Displays superblock.

a[lter] <block> <offset> <hex string>

where:

<block>	block number (decimal)
<offset>	offset within block (hex)
<hex string>	string of hex digits

Alters disk data. <hex string> should contain an even number of digits.

b[map]

Displays block map xtree copy.

d[isplay] [<block> [<offset> [<format>[<count>]]]]

<block>	block number (decimal)
<offset>	offset within block (hex)
<format>	format in which to display data (see below)
<count>	number of objects to display (decimal)

Displays data in a variety of formats.

Format may be one of the following:

a	ascii

s snapshot segment header
t snapshot table page
x xtree page

h[elp] [<command>]

<command> command name

Provides help on subcommands.

q[uit] Exits fsdb.

st[able] [<block number>]
where:

<block number> block number (decimal)

Displays summary snapshot table.

s[map] [<block number>]
where:

<block number> block number (decimal)

Displays snapshot bit map.

su[perblock]

Displays superblock.

Security

Attention RBAC users and Trusted AIX users: This command can perform privileged operations. Only privileged users can run privileged operations. For more information about authorizations and privileges, see Privileged Command Database in *Security*. For a list of privileges and the authorizations associated with this command, see the **/etc/security/privcmds** database file.

Examples

The following examples show subcommands you can use after starting the **fsdb** command on a JFS file system.

1. To display an i-node, enter:

   ```
   386i
   ```

   This command displays i-node 386 in i-node format. It now becomes the current i-node.

2. To change the link count for the current i-node to a value of 4, enter:
   ```
   ln=4
   ```
3. To increase the link count of the current i-node by a value of 1, enter:
   ```
   ln=+1
   ```
4. To display part of the file associated with the current i-node, enter:
   ```
   fc
   ```

   This command displays block 0 of the file associated with the current i-node in ASCII bytes.
5. To display entries of a directory, enter:
   ```
   2i.fd
   ```

   This changes the current i-node to the root i-node (i-node 2) and then displays the directory entries in the first block associated with that i-node. One or more of the last entries displayed may have an i-node number of 0 (zero). These are unused directory blocks; such entries cannot be manipulated as in the next example.
6. To go down a level of the directory tree, enter:
   ```
   d5i.fc
   ```

   This command changes the current i-node to the one associated with directory entry 5. Then it displays the first block of the file as ASCII text (fc). Directory entries are numbered starting from 0.
7. To display a block when you know its block number, enter:
   ```
   1b.p0o
   ```

   This command displays the superblock (block 1) of file system in octal.
8. To change the i-node of a directory entry, enter:
   ```
   2i.a0b.d7=3
   ```

   This command changes the i-node of directory entry 7 in the root directory (2i) to 3. This example also shows how several operations can be combined on one line.
9. To change the file name of a directory entry, enter:
   ```
   d7.nm="chap1.rec"
   ```

   This command changes the name field of directory entry 7 to chap1.rec.
10. To display a given block of the file associated with the current i-node, enter:
    ```
    a2b.p0d
    ```

    This command displays block 2 of the current i-node as directory entries.
11. To display the content of a single indirect block at block 7, enter:
    ```
    7b. p0S
    ```

    This command displays the block numbers allocated to the i-node that has a single indirect block at block 7.
12. To display the first page of the disk map, enter:
    ```
    0M
    ```
13. To display the first 10 words of permanent block allocation map in hexadecimal, enter:
    ```
    mp1.p10x
    ```

    This command shows the allocation bit map at the current address; for example, at 0M.

The following examples show some subcommands you can use on a JFS2 file system.

Attention: Do not use JFS2 subcommands to modify a file system.

1. To display an i-node, enter:
   ```
   inode 2
   ```
   This command displays i-node 2 in i-node format.
2. To display entries of a directory, enter:
   ```
   dir 2
   ```
   This command displays the directory entries associated with i-node 2.
3. To display a block whose block number is 0x1000, enter:
   ```
   display 0x1000
   ```
   This command displays the block at file system in hexadecimal format.

Files

/usr/sbin	Contains the **fsdb** command.
/etc/filesystems	Contains information on the file systems.

Related Information

The **dfsck** command, **fsck** command.

The **dir** file, **filsys.h** file.

The **environment** miscellaneous facility.

The **read** subroutine.

The File systems in *Operating system and device management* explains file system types, management, structure, and maintenance.

The Files in *Operating system and device management* provides information on working with files.

---

fsplit Command

Purpose

Splits FORTRAN source code into separate routine files.

Syntax

fsplit [**-e** *SubprogramUnit*] ... [*File*]

Description

The **fsplit** command takes as input either a file or standard input containing FORTRAN source code and splits the input into separate routine files of the form *name***.f**, where *name* is the name of the program unit (for example, function, subroutine, block data or program).

The name for unnamed block data subprograms has the form *blkdtaNNN***.f**, where NNN is three digits and a file of this name does not already exist. For unnamed main programs the name has the form *mainNNN***.f**. If there is an error in classifying a program unit, or if *name***.f** already exists, the program unit is put in a file of the form *zzzNNN***.f**, where *zzzNNN***.f** does not already exist.

Note: The **fsplit** command assumes that the subprogram name is on the first non-comment line of the subprogram unit. Non-standard source formats can confuse the command and produce unpredictable results.

Flags

-e *SubprogramUnit* Causes only the specified subprogram units to be split into separate files. Normally each subprogram unit is split into a separate file.

The **-e** flag can be used only for named main programs and block data subprograms. If names specified via the **-e** option are not found, a diagnostic is written to standard error.

Example

The following **fsplit** command splits the subprograms `readit` and `doit` into separate files:

```
fsplit -e readit -e doit prog.f
```

Files

/usr/bin/fsplit Contains the **fsplit** command.

Related Information

The **asa** or **fpr** command, **struct** command.

---

ftp Command

Purpose

Transfers files between a local and a remote host.

Syntax

ftp [**-d**] [**-g**] [**-i**] [**-n**] [**-v**] [**-f**] [**-K**] [**-k** *realm*] [**-q**[**-C**]] [**-s**] [*HostName* [*Port*]]

Description

The **ftp** command uses the File Transfer Protocol (FTP) to transfer files between the local host and a remote host or between two remote hosts. Remote execution of the **ftp** command is not recommended.

The FTP protocol allows data transfer between hosts that use dissimilar file systems. Although the protocol provides a high degree of flexibility in transferring data, it does not attempt to preserve file attributes (such as the protection mode or modification times of a file) that are specific to a particular file system. Moreover, the FTP protocol makes few assumptions about the overall structure of a file system and does not provide or allow such functions as recursively copying subdirectories.

Note: If you are transferring files between systems and need to preserve file attributes or recursively copy subdirectories, use the **rcp** command.

Issuing Subcommands

At the `ftp>` prompt, you can enter subcommands to perform tasks such as listing remote directories, changing the current local and remote directory, transferring multiple files in a single request, creating and removing directories, and escaping to the local shell to perform shell commands. See the Subcommands section for a description of each subcommand.

If you execute the **ftp** command and do not specify the *HostName* parameter for a remote host, the **ftp** command immediately displays the `ftp>` prompt and waits for an **ftp** subcommand. To connect to a remote host, execute the **open** subcommand**.** When the **ftp** command connects to the remote host, the **ftp** command then prompts for the login name and password before displaying the `ftp>` prompt again. The **ftp** command is unsuccessful if no password is defined at the remote host for the login name.

The **ftp** command interpreter, which handles all subcommands entered at the `ftp>` prompt, provides facilities that are not available with most file-transfer programs, such as:

- Handling file-name parameters to **ftp** subcommands
- Collecting a group of subcommands into a single subcommand macro
- Loading macros from a **$HOME/.netrc** file

These facilities help simplify repetitive tasks and allow you to use the **ftp** command in unattended mode.

The command interpreter handles file-name parameters according to the following rules:

- If a - (hyphen) is specified for the parameter, standard input (stdin) is used for read operations and standard output (stdout) is used for write operations.
- If the preceding check does not apply and file-name expansion is enabled (see the **-g** flag or the **glob** subcommand), the interpreter expands the file name according to the rules of the C shell. When globbing is enabled and a pattern-matching character is used in a subcommand that expects a single file name, results may be different than expected.

  For example, the **append** and **put** subcommands perform file-name expansion and then use only the first file name generated. Other **ftp** subcommands, such as **cd**, **delete**, **get**, **mkdir**, **rename**, and **rmdir**, do not perform file-name expansion and take the pattern-matching characters literally.
- For the **get**, **put**, **mget**, and **mput** subcommands, the interpreter has the ability to translate and map between different local and remote file-name syntax styles (see the **case**, **ntrans**, and **nmap** subcommands) and the ability to modify a local file name if it is not unique (see the **runique** subcommand). Additionally, the **ftp** command can send instructions to a remote **ftpd** server to modify a remote file name if it is not unique (see the **sunique** subcommand).
- Use double quotes (″ ″) to specify parameters that include blank characters.

> **Note:** The **ftp** command interpreter does not support pipes. It also does not necessarily support all multibyte-character file names.

To end an **ftp** session when you are running interactively, use the **quit** or **bye** subcommand or the End of File (Ctrl-D) key sequence at the `ftp>` prompt. To end a file transfer before it has completed, press the Interrupt key sequence. The default Interrupt key sequence is Ctrl-C. The **stty** command can be used to redefine this key sequence.

The **ftp** command normally halts transfers being sent (from the local host to the remote host) immediately. The **ftp** command halts transfers being received (from the remote host to the local host) by sending an FTP ABOR instruction to the remote FTP server and discarding all incoming file transfer packets until the remote server stops sending them. If the remote server does not support the ABOR instruction, the **ftp** command does not display the `ftp>` prompt until the remote server has sent all of the requested file. Additionally, if the remote server does something unexpected, you may need to end the local **ftp** process.

Security and Automatic Login

If Standard is the current authentication method:: The **ftp** command also handles security by sending passwords to the remote host and permits automatic login, file transfers, and logoff.

If you execute the **ftp** command and specify the host name (*HostName*) of a remote host, the **ftp** command tries to establish a connection to the specified host. If the **ftp** command connects successfully, the **ftp** command searches for a local **$HOME/.netrc** file in your current directory or home directory. If the

file exists, the **ftp** command searches the file for an entry initiating the login process and command macro definitions for the remote host. If the **$HOME/.netrc** file or automatic login entry does not exist or if your system has been secured with the **securetcpip** command, the **ftp** command prompts the user for a user name and password. The command displays the prompt whether or not the *HostName* parameter is specified on the command line.

Note: The queuing system does not support multibyte host names.

If the **ftp** command finds a **$HOME/.netrc** automatic login entry for the specified host, the **ftp** command attempts to use the information in that entry to log in to the remote host. The **ftp** command also loads any command macros defined in the entry. In some cases (for example, when the required password is not listed in an automatic login entry), the **ftp** command prompts for the password before displaying the `ftp>` prompt.

Once the **ftp** command completes the automatic login, the **ftp** command executes the **init** macro if the macro is defined in the automatic login entry. If the **init** macro does not exist or does not contain a **quit** or **bye** subcommand, the **ftp** command then displays the `ftp>` prompt and waits for a subcommand.

Note: The remote user name specified either at the prompt or in a **$HOME/.netrc** file must exist and have a password defined at the remote host. Otherwise, the **ftp** command fails.

If Kerberos 5 is the current authentication method:: The **ftp** command will use the extensions to ftp specifications as defined in IETF draft document ″draft-ietf-cat-ftpsec-09.txt″. The FTP security extensions will be implemented using the Generic Security Service API (GSSAPI) security mechanism. The GSSAPI provides services independent to the underlying security and communication mechanism. The GSSAPI is defined in rfc 1508 and 1509.

The **ftp** command will use the AUTH and ADAT commands to authenticate with the **ftpd** daemon. If both support Kerberos authentication, then they will use the local users DCE credentials to authenticate the user on the remote system. If this fails and Standard authentication is configured on both systems, the process described above will be used.

The *HostName* parameter is the name of the host machine to which files are transferred. The optional *Port* parameter specifies the ID of the port through which to transmit. (The **/etc/services** file specifies the default port.)

Transport Layer Security support: The **ftp** command supports Transport Layer Security (TLS) as defined in RFC 4217. TLS is a cryptographic protocol that provides secure communication between clients and servers.

The **ftp** command uses the **AUTH TLS** and **PROT P** commands to secure the communication with the **ftpd** daemon. If both the **AUTH TLS** and **PROT P** commands support the TLS protocol then a secure channel is established. Only the Standard Authentication method is supported.

When running the **ftp** command, if the **-s** flag is specified then the **ftp** command searches for a local **$HOME/.ftpcnf** file in the user's home directory. If the file is found, the **ftp** command uses the following configuration parameters to set up a TLS session with the server. If the file is not found or the configuration parameters are missing, the **ftp** command attempts to connect to the server without using the configuration parameters.

CRL_PATH
: The *CRL_PATH* parameter provides the path to the certificate revocation list file, which must be in PEM format. If specified, the digital certificate provided by the server will be verified against the certificate revocation list. If the certificate has been revoked, the TLS session will fail. If not specified, the digital certificate will not be verified against a certificate revocation list.

CA_PATH	The *CA_PATH* parameter provides the path to the certificate authority file, which must be in PEM format. If specified, the server certificate will be verified against the certificate authority. If the digital certificate provided by the server has not been signed by the security authority, the TLS session will fail. If not specified, the digital certificate provided by the server will not be verified against a certificate revocation list.
CIPHER_LIST	If the *CIPHER_LIST* parameter is specified, the list is used during the TLS session. If not, a default cipher list is used.
DEPHT	If the *CA_PATH* configuration parameter has been specified, the *DEPTH* value is used to verify the certificate provided by the ftpd server in the digital certificate hierarchy. If not provided, a default value of 9 is used.
CERTIFICATE	The *CERTIFICATE* parameter provides a path to a valid digital certificate chain file in PEM format. This file is used in the TLS session.
CERTIFICATE_PRIVATE_KEY	The *CERTIFICATE_PRIVATE_KEY* parameter contains the path to the certificate private key, in PEM format, which is used during the TLS session.

In order to support TLS, you must install the openssl-0.9.7l-1.aix5.1.ppc.rpm tool from the following Web site:
www-03.ibm.com/servers/aix/products/aixos/linux/download.html.

For Trusted AIX system:

The user is assigned a default login Sensitivity Label (SL) and Integrity Label (TL), which is the effective SL and effective TL of the user's process after a successful login. If the user does not want to log in at the default login SL, the user can choose to supply a different SL at the login time by using the **-e** option. The SL supplied by the user must be dominated by the user's clearance and contained in the system accreditation range. The TL cannot be specified by the user at login time. The default login SL and TL are defined in the **/etc/security/user** file along with the user name and the clearance for each user. To use the **-e** option, the server side's kernel trusted network bit must be turned off.

Note: Any user with user ID less than or equal to 128 cannot log in to the remote Trusted AIX system.

Flags

-C Allows the user to specify that the outgoing file sent using the **send_file** command must be cached in the Network Buffer Cache (NBC). This flag cannot be used unless the **-q** flag is specified. This flag is only applicable when a file is being sent out in the binary mode with no protection.

-d Sends debugging information about **ftp** command operations to the **syslogd** daemon. If you specify the **-d** flag, you must edit the **/etc/syslog.conf** file and add one of the following entries:

```
user.info FileName
```

OR

```
user.debug FileName
```

Note: The **syslogd** daemon debug level includes info level messages.

If you do not edit the **/etc/syslog.conf** file, no messages are produced. After changing the **/etc/syslog.conf** file, run the **refresh -s syslogd** or **kill -1** *SyslogdPID* command to inform the **syslogd** daemon of the changes to its configuration file. For more information about debug levels, refer to the **/etc/syslog.conf** file. Also, refer to the **debug** subcommand**.**

-f Causes the credentials to be forwarded. This flag will be ignored if Kerberos 5 is not the current authentication method.

-g Disables the expansion of metacharacters in file names. Interpreting metacharacters can be referred to as expanding (sometimes called globbing) a file name. See the **glob** subcommand.

-i Turns off interactive prompting during multiple file transfers. See the **prompt**, **mget**, **mput**, and **mdelete** subcommands for descriptions of prompting during multiple file transfers.

-K Disables the SO_KEEPALIVE option defined in the **sys/socket.h** file on both the control and data connection.

-k *realm* Allows the user to specify the realm of the remote station if it is different from the local systems realm. For these purposes, a realm is synonymous with a DCE cell. This flag will be ignored if Kerberos 5 is not the current authentication method.

-n Prevents an automatic login on the initial connection. Otherwise, the **ftp** command searches for a **$HOME/.netrc** entry that describes the login and initialization process for the remote host. See the **user** subcommand**.**

-q Allows the user to specify that the **send_file** subroutine must be used for sending the file on the network. This flag is only applicable when a file is being sent out in the binary mode with no protection.

-v Displays all the responses from the remote server and provides data transfer statistics. This display mode is the default when the output of the **ftp** command is to a terminal, such as the console or a display.

If stdin is not a terminal, the **ftp** command disables verbose mode unless the user invoked the **ftp** command with the **-v** flag or issued the **verbose** subcommand.

-s Starts a TLS session with the server by sending an **AUTH TLS** command and a **PROT P** command to the **ftpd** daemon. If the TLS session is established, and the user has been authenticated using the Standard Authentication method, the transfer of the data and commands is encrypted.

Subcommands

The following **ftp** subcommands can be entered at the `ftp>` prompt. Use double quotes (″ ″) to specify parameters that include blank characters.

![*Command* [*Parameters*]] Invokes an interactive shell on the local host. An optional command, with one or more optional parameters, can be given with the shell command.

$*Macro* [*Parameters*] Executes the specified macro, previously defined with the **macdef** subcommand. Parameters are not expanded.

?[*Subcommand*] Displays a help message describing the subcommand. If you do not specify a *Subcommand* parameter, the **ftp** command displays a list of known subcommands.

account [*Password*] Sends a supplemental password that a remote host may require before granting access to its resources. If the password is not supplied with the command, the user is prompted for the password. The password is not displayed on the screen.

append *LocalFile* [*RemoteFile*] Appends a local file to a file on the remote host. If the remote file name is not specified, the local file name is used, altered by any setting made with the **ntrans** subcommand or the **nmap** subcommand. The **append** subcommand uses the current values for **form**, **mode**, **struct**, and **type** subcommands while appending the file.

ascii Synonym for the **type ascii** subcommand.

bell Sounds a bell after the completion of each file transfer.

binary Synonym for the **type binary** subcommand.

block Synonym for the **mode block** subcommand.

bye Ends the file-transfer session and exits the **ftp** command. Same as the **quit** subcommand.

carriage-control Synonym for the **form carriage-control** subcommand.

case Sets a toggle for the case of file names. When the **case** subcommand is On, the **ftp** command changes remote file names displayed in all capital letters from uppercase to lowercase when writing them in the local directory. The default is Off (so the **ftp** command writes uppercase remote file names in uppercase in the local directory).

cd *RemoteDirectory* Changes the working directory on the remote host to the specified directory.

cdup Changes the working directory on the remote host to the parent of the current directory.

close Ends the file-transfer session, but does not exit the **ftp** command. Defined macros are erased. Same as the **disconnect** subcommand.

copylocal Toggles local copy. **copylocal** defaults to off. An effort is made by ftp to make sure you do not zero out a file by ftp'ing it to itself (eg. same hostname, same pathname). Turning **copylocal** ON bypasses this check.

cr Strips the carriage return character from a carriage return and line-feed sequence when receiving records during ASCII-type file transfers. (The **ftp** command terminates each ASCII-type record with a carriage return and line feed during file transfers.)

Records on remote hosts with operating systems other than the one you are running can have single line feeds embedded in records. To distinguish these embedded line feeds from record delimiters, set the **cr** subcommand to Off. The **cr** subcommand toggles between On and Off.

debug [**0** | **1**] Toggles debug record keeping On and Off. Specify **debug** or **debug 1** to print each command sent to the remote host and save the restart control file. Specify **debug** again, or **debug 0**, to stop the debug record keeping. The Ctrl-C key sequence also saves the restart control file.

Specifying the **debug** subcommand sends debugging information about **ftp** command operations to the **syslogd** daemon. If you specify the **debug** subcommand, you must edit the **/etc/syslog.conf** file and add one of the following entries:

```
user.info FileName
```

OR

```
user.debug FileName
```

Note: The **syslogd** daemon debug level includes info level messages.

If you do not edit the **/etc/syslog.conf** file, no messages are produced. After changing the **/etc/syslog.conf** file, run the **refresh -s syslogd** or **kill -1** *SyslogdPID* command to inform the **syslogd** daemon of the changes to its configuration file. For more information about debug levels, refer to the **/etc/syslog.conf** file. Also, refer to the **ftp -d** flag.

delete *RemoteFile* Deletes the specified remote file.

dir [*RemoteDirectory*][*LocalFile*] Writes a listing of the contents of the specified remote directory (*RemoteDirectory*) to the specified local file (*LocalFile*). If the *RemoteDirectory* parameter is not specified, the **dir** subcommand lists the contents of the current remote directory. If the *LocalFile* parameter is not specified or is a - (hyphen), the **dir** subcommand displays the listing on the local terminal.

disconnect Ends the file-transfer session but does not exit the **ftp** command. Defined macros are erased. Same as the **close** subcommand.

ebcdic Synonym for the **type ebcdic** subcommand.

exp_cmd Toggles between conventional and experimental protocol commands. The default is off.

file Synonym for the **struct file** subcommand.

form [**carriage-control** | **non-print** | **telnet**] Specifies the form of the file transfer. The **form** subcommand modifies the **type** subcommand to send the file transfer in the indicated form. Valid arguments are **carriage-control**, **non-print**, and **telnet**.

carriage-control
Sets the form of the file transfer to carriage-control.

non-print
Sets the form of the file transfer to non-print.

telnet Sets the form of the file transfer to Telnet. Telnet is a Transmission Control Protocol/Internet Protocol (TCP/IP) protocol that opens connections to a system.

get *RemoteFile* [*LocalFile*] Copies the remote file to the local host. If the *LocalFile* parameter is not specified, the remote file name is used locally and is altered by any settings made by the **case**, **ntrans**, and **nmap** subcommands**.** The **ftp** command uses the current settings for the **type**, **form**, **mode**, and **struct** subcommands while transferring the file.

glob
Toggles file-name expansion (globbing) for the **mdelete**, **mget**, and **mput** subcommands. If globbing is disabled, file-name parameters for these subcommands are not expanded. When globbing is enabled and a pattern-matching character is used in a subcommand that expects a single file name, results may be different than expected.

For example, the **append** and **put** subcommands perform file-name expansion and then use only the first file name generated. Other **ftp** subcommands, such as **cd**, **delete**, **get**, **mkdir**, **rename**, and **rmdir**, do not perform file-name expansion and take the pattern-matching characters literally.

Globbing for the **mput** subcommand is done locally in the same way as for the **csh** command. For the **mdelete** and **mget** subcommands, each file name is expanded separately at the remote machine and the lists are not merged. The expansion of a directory name can be different from the expansion of a file name, depending on the remote host and the **ftp** server.

To preview the expansion of a directory name, use the **mls** subcommand:

```
mls RemoteFile
```

To transfer an entire directory subtree of files, transfer a **tar** archive of the subtree in binary form, rather than using the **mget** or **mput** subcommand.

hash
Toggles hash sign (#) printing. When the **hash** subcommand is on, the **ftp** command displays one hash sign for each data block (1024 bytes) transferred.

help [*Subcommand*]
Displays help information. See the **?** subcommand.

image
Synonym for the **type image** subcommand.

lcd [*Directory*]
Changes the working directory on the local host. If you do not specify a directory, the **ftp** command uses your home directory.

local *M*
Synonym for the **type local** *M* subcommand.

ls [*RemoteDirectory*] [*LocalFile*]
Writes an abbreviated file listing of a remote directory to a local file. If the *RemoteDirectory* parameter is not specified, the **ftp** command lists the current remote directory. If the *LocalFile* parameter is not specified or is a - (hyphen), the **ftp** command displays the listing on the local terminal.

macdef *Macro*
Defines a subcommand macro. Subsequent lines up to a null line (two consecutive line feeds) are saved as the text of the macro. Up to 16 macros, containing at most 4096 characters for all macros, can be defined. Macros remain defined until either redefined or a **close** subcommand is executed.

The $ (dollar sign) and \ (backslash) are special characters in **ftp** macros. A $ symbol followed by one or more numbers is replaced by the corresponding macro parameter on the invocation line (see the **$** subcommand). A $ symbol followed by the letter i indicates that the macro is to loop, with the $i character combination being replaced by consecutive parameters on each pass.

The first macro parameter is used on the first pass, the second parameter is used on the second pass, and so on. A \ symbol prevents special treatment of the next character. Use the \ symbol to turn off the special meanings of the $ and \. (backslash period) symbols.

mdelete *RemoteFiles*
Expands the files specified by the *RemoteFiles* parameter at the remote host and deletes the remote files.

mdir [*RemoteDirectories LocalFile*]
Expands the directories specified by the *RemoteDirectories* parameter at the remote host and writes a listing of the contents of those directories to the file specified in the *LocalFile* parameter. If the *RemoteDirectories* parameter contains a pattern-matching character, the **mdir** subcommand prompts for a local file if none is specified. If the *RemoteDirectories* parameter is a list of remote directories separated by blanks, the last argument in the list must be either a local file name or a - (hyphen).

If the *LocalFile* parameter is - (hyphen), the **mdir** subcommand displays the listing on the local terminal. If interactive prompting is on (see the **prompt** subcommand), the **ftp** command prompts the user to verify that the last parameter is a local file and not a remote directory.

mget *RemoteFiles*
Expands the *RemoteFiles* parameter at the remote host and copies the indicated remote files to the current directory on the local host. See the **glob** subcommand for more information on file-name expansion. The remote file names are used locally and are altered by any settings made by the **case**, **ntrans**, and **nmap** subcommands. The **ftp** command uses the current settings for the **form**, **mode**, **struct**, and **type** subcommands while transferring the files.

mkdir [*RemoteDirectory*]
Creates the directory specified in the *RemoteDirectory* parameter on the remote host.

mls [*RemoteDirectories LocalFile*]
Expands the directories specified in the *RemoteDirectories* parameter at the remote host and writes an abbreviated file listing of the indicated remote directories to a local file. If the *RemoteDirectories* parameter contains a pattern-matching character, the **mls** subcommand prompts for a local file if none is specified. If the *RemoteDirectories* parameter is a list of remote directories separated by blanks, the last argument in the list must be either a local file name or a - (hyphen).

If the *LocalFile* parameter is - (hyphen), the **mls** subcommand displays the listing on the local terminal. If interactive prompting is on (see the **prompt** subcommand), the **ftp** command prompts the user to verify that the last parameter is a local file and not a remote directory.

mode [**stream** | **block**]
Sets file-transfer mode. If an argument is not supplied, the default is **stream**.

block Sets the file-transfer mode to block.

stream Sets the file-transfer mode to stream.

modtime
Shows the last modification time of the specified file on the remote machine. If the **ftp** command is not connected to a host prior to execution, the **modtime** subcommand terminates with an error message. The **ftp** command ignores parameter beyond the first parameter. If the *FileName* parameter is not specified, the **ftp** command prompts for a file name. If no file name is given, the **ftp** command sends a usage message to standard output and terminates the subcommand.

If the name specified by the *FileName* parameter exists on the remote host, and the name specifies a file, then the **ftp** command sends a message containing the last modification time of the file to standard output and terminates the subcommand. If *FileName* specifies a directory, the **ftp** command sends an error message to standard output and terminates the subcommand.

Note: The **modtime** subcommand interprets metacharacters when allowed.

mput [*LocalFiles*]
Expands the files specified in the *LocalFiles* parameter at the local host and copies the indicated local files to the remote host. See the **glob** subcommand for more information on file-name expansion. The local file names are used at the remote host and are altered by any settings made by the **ntrans** and **nmap** subcommands. The **ftp** command uses the current settings for the **type**, **form**, **mode**, and **struct** subcommands while transferring the files.

nlist [*RemoteDirectory*][*LocalFile*]
Writes a listing of the contents of the specified remote directory (*RemoteDirectory*) to the specified local file (*LocalFile*). If the *RemoteDirectory* parameter is not specified, the **nlist** subcommand lists the contents of the current remote directory. If the *LocalFile* parameter is not specified or is a - (hyphen), the **nlist** subcommand displays the listing on the local terminal.

nmap [*InPattern OutPattern*]

Turns the file-name mapping mechanism On or Off. If no parameters are specified, file-name mapping is turned off. If parameters are specified, source file names are mapped for the **mget** and **mput** subcommands and for the **get** and **put** subcommands when the destination file name is not specified. This subcommand is useful when the local and remote hosts use different file-naming conventions or practices. Mapping follows the pattern set by the *InPattern* and *OutPattern* parameters.

The *InPattern* parameter specifies the template for incoming file names, which may have already been processed according to the **case** and **ntrans** settings. The template variables `$1` through `$9` can be included in the *InPattern* parameter. All characters in the *InPattern* parameter, other than the $ (dollar sign) and the \$ (backslash, dollar sign), are treated literally and are used as delimiters between *InPattern* variables. For example, if the *InPattern* parameter is `$1.$2` and the remote file name is `mydata.dat`, the value of `$1` is `mydata` and the value of `$2` is `dat`.

The *OutPattern* parameter determines the resulting file name. The variables `$1` through `$9` are replaced by their values as derived from the *InPattern* parameter, and the variable `$0` is replaced by the original file name. Additionally, the sequence [*Sequence1*,*Sequence2*] is replaced by the value of *Sequence1*, if *Sequence1* is not null; otherwise, it is replaced by the value of *Sequence2*. For example, the subcommand:

```
nmap $1.$2.$3 [$1,$2].[$2,file]
```

would yield `myfile.data` from `myfile.data` or `myfile.data.old`, `myfile.file` from `myfile`, and `myfile.myfile` from `.myfile`. Use the \ (backslash) symbol to prevent the special meanings of the $ (dollar sign), [(left bracket),] (right bracket), and , (comma) in the *OutPattern* parameter.

non-print

Synonym for the **form non-print** subcommand.

ntrans [*InCharacters* [*OutCharacters*]]

Turns the file-name character translation mechanism On and Off. If no parameters are specified, character translation is turned off. If parameters are specified, characters in source file names are translated for **mget** and **mput** subcommands and for **get** and **put** subcommands when the destination file name is not specified.

This subcommand is useful when the local and remote hosts use different file-naming conventions or practices. Character translation follows the pattern set by the *InCharacters* and *OutCharacters* parameter. Characters in a source file name matching characters in the *InCharacters* parameter are replaced by the corresponding characters in the *OutCharacters* parameter.

If the string specified by the *InCharacters* parameter is longer than the string specified by the *OutCharacters* parameter, the characters in the *InCharacters* parameter are deleted if they have no corresponding character in the *OutCharacters* parameter.

open *HostName* [*Port*]

Establishes a connection to the FTP server at the host specified by the *HostName* parameter. If the optional port number is specified, the **ftp** command attempts to connect to a server at that port. If the automatic login feature is set (that is, the **-n** flag was not specified on the command line), the **ftp** command attempts to log in the user to the FTP server.

You must also have a **$HOME/.netrc** file with the correct information in it and the correct permissions set. The **.netrc** file must be in your home directory.

passive

Toggles passive mode for file transfers. When a file transfer command (such as **get**, **mget**, **put**, or **mput**) is invoked with passive mode off, the **ftp** server opens a data connection back to the client. In passive mode, the client opens data connections to the server when sending or receiving data.

private

Sets the protection level to ″private.″ At this level, data is integrity and confidentially protected.

prompt

Toggles interactive prompting. If interactive prompting is on (the default), the **ftp** command prompts for verification before retrieving, sending, or deleting multiple files during the **mget**, **mput**, and **mdelete** subcommands. Otherwise, the **ftp** command acts accordingly on all files specified.

protect
This command returns the current level of protection.

proxy [*Subcommand*]
Executes an **ftp** command on a secondary control connection. This subcommand allows the **ftp** command to connect simultaneously to two remote FTP servers for transferring files between the two servers. The first **proxy** subcommand should be an **open** subcommand to establish the secondary control connection. Enter the **proxy ?** subcommand to see the other **ftp** subcommands that are executable on the secondary connection.

The following subcommands behave differently when prefaced by the **proxy** subcommand:

- The **open** subcommand does not define new macros during the automatic login process.
- The **close** subcommand does not erase existing macro definitions.
- The **get** and **mget** subcommands transfer files from the host on the primary connection to the host on the secondary connection.
- The **put**, **mput**, and **append** subcommands transfer files from the host on the secondary connection to the host on the primary connection.
- The **restart** subcommand can be handled by the **proxy** command.
- The **status** subcommand displays accurate information.

File transfers require that the FTP server on the secondary connection must support the PASV (passive) instruction.

put *LocalFile* [*RemoteFile*]
Stores a local file on the remote host. If you do not specify the *RemoteFile* parameter, the **ftp** command uses the local file name to name the remote file, and the remote file name is altered by any settings made by the **ntrans** and **nmap** subcommands. The **ftp** command uses the current settings for the **type**, **form**, **mode**, and **struct** subcommands while transferring the files.

pwd
Displays the name of the current directory on the remote host.

quit
Closes the connection and exits the **ftp** command. Same as the **bye** subcommand.

quote *String*
Sends the string specified by the *String* parameter verbatim to the remote host. Execute the **remotehelp** or **quote help** subcommand to display a list of valid values for the *String* parameter.

> **Note:** ″Quoting″ commands that involve data transfers can produce unpredictable results.

record
Synonym for the **struct record** subcommand.

recv *RemoteFile* [*LocalFile*]
Copies the remote file to the local host. Same as the **get** subcommand.

reinitialize
Reinitializes an FTP session by flushing all I/O and allowing transfers to complete. Resets all defaults as if a user had just started an FTP session without logging in to a remote host.

remotehelp [*Subcommand*]
Requests help from the remote FTP server.

rename *FromName ToName*
Renames a file on the remote host.

reset
Clears the reply queue. This subcommand resynchronizes the command parsing.

restart get | **put** | **append**
Restarts a file transfer at the point where the last checkpoint was made. To run successfully, the subcommand must be the same as the aborted subcommand, including structure, type, and form. Valid arguments are **get**, **put**, and **append**.

rmdir *RemoteDirectory*
Removes the remote directory specified by the *RemoteDirectory* parameter at the remote host.

runique (ReceiveUnique) Toggles the facility for creating unique file names for local destination files during **get** and **mget** subcommands. If this facility is Off (the default), the **ftp** command overwrites local files. Otherwise, if a local file has the same name as that specified for a local destination file, the **ftp** command modifies the specified name of the local destination file with **.**1. If a local file is already using the new name, the **ftp** command appends the postfix .2 to the specified name. If a local file is already using this second name, the **ftp** command continues incrementing the postfix until it either finds a unique file name or reaches .99 without finding a unique file name. If the **ftp** command cannot find a unique file name, the **ftp** command reports an error and the transfer does not take place. Note that the **runique** subcommand does not affect local file names generated from a shell command.

safe Sets the protection level to ″safe.″ At this level, data is integrity protected.

send *LocalFile* [*RemoteFile*] Stores a local file on the remote host. Same as the **put** subcommand.

sendport Toggles the use of FTP PORT instructions. By default, the **ftp** command uses a PORT instruction when establishing a connection for each data transfer. When the use of PORT instructions is disabled, the **ftp** command does not use PORT instructions for data transfers. The PORT instruction is useful when dealing with FTP servers that ignore PORT instructions while incorrectly indicating the instructions have been accepted.

site *Args* Displays or sets the idle time-out period, displays or sets the file-creation umask, or changes the permissions of a file, using the **chmod** command. Possible values for the *Args* parameter are **umask** and **chmod**.

size *RemoteFile* Displays the size in bytes of the remote file specified by the *RemoteFile* parameter.

status Displays the current status of the **ftp** command as well as the status of the subcommands.

stream Synonym for the **mode stream** subcommand.

struct [**file** | **record**] Sets the data transfer structure type. Valid arguments are **file** and **record**.

file Sets the data-transfer structure type to file.

record Sets the data-transfer structure type to record.

sunique (Send/Store Unique) Toggles the facility for creating unique file names for remote destination files during **put** and **mput** subcommands. If this facility is off (the default), the **ftp** command overwrites remote files. Otherwise, if a remote file has the same name as that specified for a remote destination file, the remote FTP server modifies the name of the remote destination file. Note that the remote server must support the STOU instruction.

system Shows the type of operating system running on the remote machine.

telnet Synonym for the **form telnet** subcommand.

tenex Synonym for the **type tenex** subcommand.

trace Toggles packet tracing.

type [**ascii** | **binary** | **ebcdic** | **image** | **local** *M* | **tenex**] Sets the file-transfer type. Valid arguments are **ascii**, **binary**, **ebcdic**, **image**, **local** *M*, and **tenex**. If an argument is not specified, the current type is printed. The default type is **ascii**; the **binary** type can be more efficient than **ascii**.

ascii Sets the file-transfer type to network ASCII. This type is the default. File transfer may be more efficient with binary-image transfer. See the **binary** argument for further information.

binary Sets the file-transfer type to binary image. This type can be more efficient than an ASCII transfer.

ebcdic Sets the file-transfer type to EBCDIC.

image Sets the file-transfer type to binary image. This type can be more efficient than an ASCII transfer.

local *M*
Sets the file-transfer type to local. The *M* parameter defines the decimal number of bits per machine word. This parameter does not have a default.

tenex Sets the file-transfer type to that needed for TENEX machines.

user *User* [*Password*] [*Account*]
Identifies the local user (*User*) to the remote FTP server. If the *Password* or *Account* parameter is not specified and the remote server requires it, the **ftp** command prompts for the password or account locally. If the *Account* parameter is required, the **ftp** command sends it to the remote server after the remote login process completes.

> **Note:** Unless automatic login is disabled by specifying the **-n** flag on the command line, the **ftp** command sends the *User*, *Password*, and *Account* parameters automatically for the initial connection to the remote server. You also need a **.netrc** file in your home directory in order to issue an automatic login.

verbose
Toggles verbose mode. When the verbose mode is on (the default), the **ftp** command displays all responses from the remote FTP server. Additionally, the **ftp** command displays statistics on all file transfers when the transfers complete.

Security

Attention RBAC users and Trusted AIX users: This command can perform privileged operations. Only privileged users can run privileged operations. For more information about authorizations and privileges, see Privileged Command Database in *Security*. For a list of privileges and the authorizations associated with this command, see the **/etc/security/privcmds** database file.

Examples

1. To invoke the **ftp** command, log in to the system `canopus`, display local help information, display remote help information, display status, toggle the **bell**, **prompt**, **runique**, **trace**, and **verbose** subcommands, and then quit, enter:

```
$ ftp canopus
Connected to canopus.austin.century.com.
220 canopus.austin.century.com FTP server (Version 4.1 Sat Nov 23 12:52:09 CST 1991) ready.
Name (canopus:eric): dee
331 Password required for dee.
Password:
230 User dee logged in.
ftp> help
Commands may be abbreviated. Commands are:
!          delete          mdelete          proxy          runique
$          debug           mdir             sendport       send
account    dir             mget             put            size
append     disconnect      mkdir            pwd            status
ascii      form            mls              quit           struct
bell       get             mode             quote          sunique
binary     glob            modtime          recv           system
bye        hash            mput             remotehelp     tenex
case       help            nmap             rstatus        trace
cd         image           nlist            rhelp          type
cdup       lcd             ntrans           rename         user
close      ls              open             reset          verbose
cr         macdef          prompt           rmdir          ?
clear      private         protect          safe
ftp> remotehelp
214-The following commands are recognized(* =>'s unimplemented).
 USER   PORT   RETR   MSND*  ALLO   DELE   SITE*  XMKD   CDUP
 PASS   PASV   STOR   MSOM*  REST*  CWD    STAT*  RMD    XCUP
 ACCT*  TYPE   APPE   MSAM*  RNFR   XCWD   HELP   XRMD   STOU
 REIN*  STRU   MLFL*  MRSQ*  RNTO   LIST   NOOP   PWD
 QUIT   MODE   MAIL*  MRCP*  ABOR   NLST   MKD    XPWD
 AUTH   ADAT   PROT   PBSZ   MIC    ENC    CCC
214 Direct comments to ftp-bugs@canopus.austin.century.com.
ftp> status
Connected to canopus.austin.century.com.
```

```
No proxy connection.
Mode: stream; Type: ascii; Form: non-print; Structure: file
Verbose: on; Bell: off; Prompting: on; Globbing: on
Store unique: off; Receive unique: off
Case: off; CR stripping: on
Ntrans: off
Nmap: off
Hash mark printing: off; Use of PORT cmds: on
ftp> bell
Bell mode on.
ftp> prompt
Interactive mode off.
ftp> runique
Receive unique on.
ftp> trace
Packet tracing on.
ftp> verbose
Verbose mode off.
ftp> quit
$
```

2. To invoke the **ftp** command, log in to the system `canopus`, print the working directory, change the working directory, set the file transfer type to ASCII, send a local file to the remote host, change the working directory to the parent directory, and then quit, enter:

```
$ ftp canopus
Connected to canopus.austin.century.com.
220 canopus.austin.century.com FTP server (Version 4.1 Sat Nov 23 12:52:09 CST 1991) ready.
Name (canopus:eric): dee
331 Password required for dee.
Password:
230 User dee logged in.
ftp> pwd
257 "/home/dee" is current directory.
ftp> cd desktop
250 CWD command successful.
ftp> type ascii
200 Type set to A.
ftp> send typescript
200 PORT command successful.
150 Opening data connection for typescript (128.114.4.99,1412).
226 Transfer complete.
ftp> cdup
250 CWD command successful.
ftp> bye
221 Goodbye.
$
```

3. To invoke the **ftp** command with automatic logon (using the **.netrc** file), open a session with the system `canopus`, log in, change the working directory to the parent directory, print the working directory, list the contents of the current directory, delete a file, write a listing of the contents of the current directory to a local file, close the session, and then quit, enter:

```
$ ftp canopus
Connected to canopus.austin.century.com.
220 canopus.austin.century.com FTP server (Version 4.1 Sat Nov 23 12:52:09 CST 1991) ready.
331 Password required for dee.
230 User dee logged in.
ftp> cdup
250 CWD command successful.
ftp> pwd
257 "/home" is current directory.
ftp> dir
200 PORT command successful.
```

```
150 Opening data connection for /usr/bin/ls (128.114.4.99,1407)
(0 bytes).
total 104
drwxr-xr-x   2 system        32 Feb 23 17:55 bin
Drwxr-xr-x  26 rios        4000 May 30 17:18 bin1
drwxr-xr-x   2 system        32 Feb 23 17:55 books
drwxrwxrwx  18 rios        1152 Jun  5 13:41 dee
-r--r--r--   1 system      9452 May 17 12:21 filesystems
drwxr-xr-x   2 system        32 Feb 23 17:55 jim
drwxr-xr-x   5 system        80 Feb 23 17:55 krs
drwxrwxrwx   2 rios       16432 Feb 23 17:36 lost+found
-rwxr-xr-x   1 rios        3651 May 24 16:45 oldmail
drwxr-xr-x   2 system       256 Feb 23 17:55 pubserv
drwxrwxrwx   2 system       144 Feb 23 17:55 rein989
drwxr-xr-x   2 system       112 Feb 23 17:55 reinstall
226 Transfer complete.
ftp> delete oldmail
250 DELE command successful.
ftp> mdir /home/dee/bin binlist
output to local-file: binlist? y
200 PORT command successful.
150 Opening data connection for /usr/bin/ls (128.114.4.99,1408) (0 bytes).
226 Transfer complete.
ftp> close
221 Goodbye.
ftp> quit
$
```

4. To invoke the **ftp** command to log in to the remote host icehouse.austin.ibm.com with a user name david of the specific SLs sec a b, enter:

```
$ ftp icehouse.aoot.austin.ibm.com
Connected to icehouse.aoot.austin.ibm.com.
220 icehouse.aoot.austin.ibm.com FTP server (Version 4.2 Mon Jan 29 11:26:29 CST 2007) ready.
Name (icehouse.aoot.austin.ibm.com:zdai): david -e "sec a b"
331 Password required for david.
Password:
230-Last unsuccessful login: Thu Feb 15 20:29:38 CST 2007 on /dev/pts/2 from zdai.austin.ibm.com
230-Last login: Mon Feb 19 20:41:07 CST 2007 on /dev/pts/2 from zdai.austin.ibm.com
230 User david logged in.
ftp>
```

Files

/usr/samples/tcpip/netrc	Contains the sample **.netrc** file.
/etc/syslog.conf	Contains configuration information for the **syslogd** daemon.
/usr/samples/tcpip/.ftpconf	Contains the example **.ftpcnf** file.

Related Information

The **csh** command, **kill** command, **rcp** command, **refresh** command, **rlogin** command, **rsh** command, **stty** command, **telnet** command, **tftp** command.

The **ftpd** daemon, the **syslogd** daemon.

The **.netrc** file format.

The **.ftpcnf** file format.

File transfers using the ftp and rcp commands in *Networks and communication management*.

Communications and networks in *Networks and communication management*.

Authentication and the secure rcmds in *Networks and communication management*.

Network option tunable parameters in *Performance management*.

---

ftpd Daemon

Purpose

Provides the server function for the Internet FTP protocol.

Syntax

> **Note:** The **ftpd** daemon is usually started by the **inetd** daemon. It can also be controlled from the command line, using SRC commands.

/usr/sbin/ftpd [**-d**] [**-f**] [**-ff**] [**-k**] [**-l**] [**-U**] [**-t** *TimeOut*] [**-T** *MaxTimeOut*] [**-s**] [**-u** *OctalVal*] [**-q** [**-C**]]

Description

The **/usr/sbin/ftpd** daemon is the DARPA Internet File Transfer Protocol (FTP) server process. The **ftpd** daemon uses the Transmission Control Protocol (TCP) to listen at the port specified with the **ftp** command service specification in the **/etc/services** file.

Changes to the **ftpd** daemon can be made using the System Management Interface Tool (SMIT) or System Resource Controller (SRC), by editing the **/etc/inetd.conf** or /**etc**/**services** file. Typing `ftpd` at the command line is not recommended. The **ftpd** daemon is started by default when it is uncommented in the **/etc/inetd.conf** file.

The **inetd** daemon gets its information from the /**etc**/**inetd.conf** file and the /**etc**/**services** file.

If you change the **/etc/inetd.conf** or /**etc**/**services** file, run the **refresh -s inetd** or **kill -1** *InetdPID* command to inform the **inetd** daemon of the changes to its configuration files.

The **ftpd** daemon expands file names according to the conventions of the **csh** command. This command allows you to use such metacharacters as the * (asterisk), the ? (question mark), [ ] (left and right brackets), { } (left and right braces), and the ~ (tilde).

ftpaccess.ctl File

The **/etc/ftpaccess.ctl** file is searched for lines that start with **allow:**, **deny:**, **readonly:**, **writeonly:**, **readwrite:**, **useronly:**, **grouponly:**, **herald:** and/or **motd:**. Other lines are ignored. If the file doesn't exist, then ftp access is allowed for all hosts. The **allow:** and **deny:** lines are for restricting host access. The **readonly:**, **writeonly:** and **readwrite:** lines are for restricting ftp reads (get) and writes (put). The **useronly:** and **grouponly:** lines are for defining anonymous users. The **herald:** and **motd:** lines are for multiline messages before and after login.

The syntax for all lines in **/etc/ftpaccess.ctl** is in the form:

```
keyword: value, value, ...
```

where one can specify one or more values for every keyword. One can have multiple lines with the same keyword. The lines in **/etc/ftpaccess.ctl** are limited to 1024 characters, anything more than 1024 characters will be ignored.

The syntax for the **allow:** and **deny:** lines are:

```
allow: host, host, ...
deny: host, host, ...
```

If an **allow:** line is specified, then only the hosts listed in all the **allow:** lines are allowed ftp access. All other hosts will be refused ftp access. If there is no **allow:** line, then all hosts will be given ftp access except those hosts specified in the **deny:** line(s). The host can be specified as either a hostname or IP address.

The syntax for the **readonly**:, **writeonly**: and **readwrite**: lines is:

```
readonly: dirname, dirname, ...
writeonly: dirname, dirname, ...
readwrite: dirname, dirname, ...
```

The **readonly:** lines list the read-only directories and the **writeonly:** lines list the write-only directories. Read access is denied in a write-only directory and write access is denied in a read-only directory. All other directories are granted access except when a **readwrite:** line is specified. If a **readwrite:** line is specified, only directories listed in the **readwrite:** line and/or listed in the **readonly:** line are granted access for reading, AND only directories listed in the **readwrite:** line and/or listed in the **writeonly:** line are granted access for writing. Also, these lines can have a value of ″ALL″ or ″NONE″.

The syntax for the **useronly:**, **puseronly:**, **grouponly:**, and **pgrouponly:** lines is:

```
useronly: username, username, ...
puseronly: username, username, ...
grouponly: groupname, groupname, ...
pgrouponly: groupname, groupname, ...
```

The username is from **/etc/passwd** and the groupname is from **/etc/group**. The **useronly:** and **puseronly:** lines define an anonymous user. The **grouponly:** and **pgrouponly:** lines define a group of anonymous users. These anonymous users are similar to the user anonymous in that ftp activity is restricted to their home directories. **The useronly:** and **grouponly:** lines define anonymous users similar to the user anonymous in that they are not password protected. The **puseronly:** and **pgrouponly:** lines define anonymous users that are password protected.

Note: For **puseronly:** and **pgrouponly:** users, passwords must be created and login must be disabled.

The syntax for the **herald:** and **motd:** lines are:

```
herald: path
motd: on|off
```

The path is the full path name of the file that contains the multiline herald that will be displayed before login. When the **motd:** line has a value of 'on', then the **$HOME/motd** file contains the multiline message that will displayed after login. If the user is a defined anonymous user, then the **/etc/motd** file contains the multiline message that will displayed after login. (Note that **/etc/motd** is in the anonymous user's chroot'ed home directory). The default for the **motd:** line is off.

If the current authentication method is the Standard Operating system authentication method:

Before the **ftpd** daemon can transfer files for a client process, it must authenticate the client process. The **ftpd** daemon authenticates client processes according to these rules:

- The user must have a password in the password database, **/etc/security/passwd**. (If the user's password is not null, the client process must provide that password.)
- The user name must not appear in the **/etc/ftpusers** file.
- The user's login shell must appear in the shells attribute of the **/etc/security/login.cfg** file.
- If the user name is `anonymous`, `ftp` or is a defined anonymous user in the **/etc/ftpaccess.ctl** file, an anonymous FTP account must be defined in the password file. In this case, the client process is

allowed to log in using any password. By convention, the password is the name of the client host. The **ftpd** daemon takes special measures to restrict access by the client process to the anonymous account.

If Kerberos 5 is the current authentication method:

The **ftpd** daemon allows access only if all of the following conditions are satisfied:

- The local user of the ftp client has current DCE credentials.
- The local and remote systems both support the **AUTH** command.
- The remote system accepts the DCE credentials as sufficient for access to the remote account. See the **kvalid_user** function for additional information.

Transport Layer Security support

The **ftpd** daemon supports Transport Layer Security (TLS) as defined in RFC 4217. TLS is a cryptographic protocol that provides secure communication between clients and servers.

The main purpose of the implementation is to secure the control and data connection using encryption. The client needs to be authenticated by other means. The only supported method is the Standard Authentication method.

Upon receiving a request to start a TLS session, the **ftpd** daemon proceeds to read the **/etc/ftpd.conf** file, loading the following configuration parameters that will be used to set up the TLS session:

CRL_PATH	The *CRL_PATH* parameter provides the path to the certificate revocation list file, which must be in PEM format. If specified, the digital certificate provided by the client will be verified against the certificate revocation list. If the ftp client is not using a digital certificate, the connection will fail. If the client provides a digital certificate, but the certificate has been revoked, the TLS session will fail. If this parameter is not specified, the client does not have to provide a digital certificate.
CA_PATH	The *CA_PATH* parameter provides the path to the certificate authority file, which must be in PEM format. If specified, the client certificate will be verified against the certificate authority. If the client does not provide a digital certificate, the connection will fail. If the client provides a digital certificate, but the certificate has not been signed by the security authority, the TLS session will fail. If this parameter is not specified, the client does not have to provide a digital certificate.
CIPHER_LIST	If the *CIPHER_LIST* parameter is specified, the list is used during the TLS session. If not, a default cipher list is used.
DEPHT	If the *CA_PATH* configuration parameter has been specified, the *DEPTH* value is used to verify the certificate provided by the ftp client in the digital certificate hierarchy. If not provided, a default value of 9 is used.
CERTIFICATE	The *CERTIFICATE* parameter provides a path to a valid digital certificate chain file in PEM format. This file is used in the TLS session. This parameter needs to be specified in order to start a TLS session. If this parameter is not specified, the ftpd server rejects all TLS requests.
CERTIFICATE_PRIVATE_KEY	The *CERTIFICATE_PRIVATE_KEY* parameter provides the path to the certificate private key, which is in PEM format, and is used during the TLS session. This parameter needs to be specified in order to start a TLS session. If this parameter is not specified, the ftpd server rejects all TLS requests.
DH_PARAMETERS_DIR	The *DH_PARAMETERS_DIR* parameter provides the path to a directory containing *Diffie Helman* parameters in PEM format. More than one file containing *Diffie Helman* parameters in PEM format can be included in this directory. The **ftpd** daemon searches for the appropriate parameter to use if required.

In order to support TLS, you must install the openssl-0.9.7l-1.aix5.1.ppc.rpm tool from the following Web site:
www-03.ibm.com/servers/aix/products/aixos/linux/download.html.

File Transfer Protocol Subtree Guidelines

When handling an anonymous FTP user, the server performs the **chroot** command in the home directory of the FTP user account. For greater security, implement the following rules when you construct the FTP subtree:

~ftp Make the home directory owned by root and mode r-xr-xr-x (555).

~ftp/bin Make this directory owned by the root user and not writable by anyone. The **ls** program must be present in this directory to support the list command. This program should have mode 111.

~ftp/etc Make this directory owned by the root user and not writable by anyone.

~ftp/pub Make this directory mode 777 and owned by FTP. Users should then place files that are to be accessible through the anonymous account in this directory.

Note: The shell script **/usr/samples/tcpip/anon.ftp** uses the above rules to set up the anonymous FTP account for you.

When handling an anonymous FTP user defined in **/etc/ftpaccess.ctl**, the server performs the **chroot** command in the home directory of the user account. For greater security, implement the following rules when you construct the user's subtree:

~user Make the home directory owned by root and mode `r-xr-xr-x` (555).

~user/bin
Make this directory owned by the root user and unwritable by anyone. The **ls** program must be present in this directory to support the list command. This program should have mode 111.

~user/etc
Make this directory owned by the root user and unwritable by anyone.

~user/pub
Make this directory mode 777 and owned by user. Users should then place files that are to be accessible through the `anonymous` account in this directory.

Note: The shell script **/usr/samples/tcpip/anon.users.ftp** uses the above rules to set up the anonymous FTP account for you.

The server must run as the root user to create sockets with privileged port numbers. The server maintains an effective user ID of the logged-in user, reverting to the root user only when binding addresses to sockets.

Supported File Transfer Protocol Requests

The **ftpd** daemon currently supports the following FTP requests:

ABOR Terminates previous command.
ACCT Specifies account (ignored).
ADAT Specifies the Authentication/Security Data.
ALLO Allocates storage (vacuously).
APPE Appends to a file.
AUTH Specifies the Authentication/Security Mechanism.
CCC Specifies the Clear Command Channel.
CDUP Changes to the parent directory of the current working directory.
CWD Changes working directory.
DELE Deletes a file.
ENC Specifies the Privacy Protected Command.

HELP Gives help information.

LIST Gives list files in a directory (this FTP request is the same as the **ls -lA** command).
MKD Makes a directory.
MDTM Shows last modification time of file.
MIC Specifies the Integrity Protected Command.
MODE Specifies data transfer mode.
NLST Gives a name list of files in directory (this FTP request is the same as the **ls** command).
NOOP Does nothing.
PASS Specifies a password.
PASV Prepares for server-to-server transfers.
PBSZ Specifies the Protection Buffer Size.
PORT Specifies a data connection port.
PROT Specifies the Data Channel Protection Level.
PWD Prints the current working directory.
QUIT Terminates session.
RETR Retrieves a file.
RMD Removes a directory.
RNFR Specifies rename-from file name.
RNTO Specifies rename-to file name.
SITE The following nonstandard or UNIX-specific commands are supported by the **SITE** request:

UMASK
Changes umask (**SITE UMASK 002**).

IDLE Sets idler time (**SITE IDLE 60**).

CHMOD
Changes mode of a file (**SITE CHMOD 755** *FileName*).

HELP Gives help information (**SITE HELP**).

SIZE Returns size of current file.
STAT Returns the status of the server.
STOR Stores a file.
STOU Stores a file using a unique file name.
STRU Specifies the structure of data transfer as a file structure.
SYST Shows operating system type of server system.
TYPE Specifies data transfer type with the *Type* parameter.
USER Specifies user name.
XCUP Changes the parent directory of the current working directory (not usually used).
XCWD Changes current directory (not usually used).
XMKD Creates a directory (not usually used).
XPWD Prints the current working directory (not usually used).
XRMD Removes a directory (not usually used).

The remaining FTP requests defined in Internet RFC 959 are recognized, but not implemented. The **MDTM** and **SIZE** requests are not specified by RFC 959, but are scheduled to appear in the next updated FTP RFC.

If a **STAT** request is received during a data transfer and preceded by both a Telnet **IP** signal and **SYNCH** signal, transfer status is returned.

The **ftpd** daemon should be controlled using the System Management Interface Tool (SMIT) or by changing the **/etc/inetd.conf** file. Typing `ftpd` at the command line is not recommended.

Manipulating the ftpd Daemon with the System Resource Controller

The **ftpd** daemon is a subserver of the **inetd** daemon, which is a subsystem of the System Resource Controller (SRC). The **ftpd** daemon is a member of the **tcpip** SRC subsystem group. This daemon is enabled by default in the **/etc/inetd.conf** file and can be manipulated by the following SRC commands:

startsrc	Starts a subsystem, group of subsystems, or a subserver.
stopsrc	Stops a subsystem, group of subsystems, or a subserver.
lssrc	Gets the status of a subsystem, group of subsystems, or a subserver.

Flags

-C Allows the user to specify that the outgoing file sent using the **send_file** command must be cached in the Network Buffer Cache (NBC). This flag cannot be used unless the **-q** flag is specified. This flag is only applicable when a file is being sent out in the binary mode with no protection.

-d Sends debugging information about **ftpd** daemon operations to the **syslogd** daemon. If you specify the **-d** flag, you must edit the **/etc/syslog.conf** file and add the following entry:

```
daemon.debug FileName
```

Note: The **syslogd** daemon's `debug` level includes `info` level messages.

If you do not edit the **/etc/syslog.conf** file, no messages are produced. After changing the **/etc/syslog.conf** file, run the **refresh -s syslogd** command or **kill -1** *SyslogdPID* command to inform the **syslogd** daemon of the changes to its configuration file. For more information about debug levels, refer to the **/etc/syslog.conf** file.

-f Disables checking for a privileged port when the client requests the server to connect back to a specific port. By default, **ftpd** does not allow the client to request a connection to a privileged port as a security precaution.

-ff Disables checking for both a privileged port and an IP address that matches the one used for the control connection when the client requests the server to connect back to a specific client port. Using this flag enables the client to request that the server send data to an alternate host or interface. By default, **ftpd** does not allow this action as a security precaution.

-k Sets the **SO_KEEPALIVE** option defined in the **sys/socket.h** file on the data transfer socket to enable the data transfer to time out in the event TCP/IP hangs. The idle interval time is based on system-wide values designated by the `tcp_keepidle` and `tcp_keepintvl` options of the **no** command. Without the flag, **ftpd** data transfer will not time out.

-l Sends logging information about **ftpd** daemon operations to the **syslogd** daemon. If you specify the **-l** flag, you must edit the **/etc/syslog.conf** file and add the following entry:

```
daemon.info FileName
```

If you do not edit the **/etc/syslog.conf** file, no messages are produced. After changing the **/etc/syslog.conf** file, run the **refresh -s syslogd** command or **kill -1** *SyslogdPID* command to inform the **syslogd** daemon of the changes to its configuration file. For more information about debug levels, refer to the **/etc/syslog.conf** file.

-q Allows the user to specify that the **send_file** subroutine must be used for sending the file on the network. This flag is only applicable when a file is being sent out in the binary mode with no protection.

-t *TimeOut* Logs out inactive sessions after the number of seconds specified by the *TimeOut* variable. The default limit is 15 minutes (900 seconds). The timeout applies to both the data and the control connections.

-T *MaxTimeOut* Logs out inactive client sessions after a maximum number of seconds specified by the *MaxTimeOut* variable. The default limit is 2 hours (7200 seconds).

-s Turns on socket-level debugging.

-u *OctalVal* Sets the **ftpd** daemon's umask. The *OctalVal* variable must be specified as an octal value to define the umask. The default umask is an octal value of 027, which results in file permissions of **rw-r-----**.

-U Keep files unlocked while in transfer. If this flag is specified with **/usr/sbin/ftpd**, then the file can be opened while still in transfer.

Security

The **ftpd** daemon is a PAM-enabled application with a service name of *ftp*. System-wide configuration to use PAM for authentication is set by modifying the value of the **auth_type** attribute, in the **usw** stanza of **/etc/security/login.cfg**, to PAM_AUTH as the root user.

The authentication mechanisms used when PAM is enabled depend on the configuration for the **ftp** service in **/etc/pam.conf**. The **ftpd** daemon requires **/etc/pam.conf** entries for the **auth**, **account**, and **session** module types. Listed below is a recommended configuration in **/etc/pam.conf** for the **ftp** service:

```
#
# AIX ftp configuration
#
ftp auth      required     /usr/lib/security/pam_aix

ftp account   required     /usr/lib/security/pam_aix

ftp session   required     /usr/lib/security/pam_aix
```

Examples

Note: The arguments for the **ftpd** daemon can be specified by using SMIT or by editing the **/etc/inetd.conf** file.

1. To start the **ftpd** daemon, type the following:

   ```
   startsrc -t ftp
   ```

   The **startsrc** command with the **-t** flag starts the **ftpd** subserver. You must use the **-t** flag to specify a subserver. Otherwise, the command does not execute properly.
2. To stop the **ftpd** daemon, usually type the following:

   ```
   stopsrc -t ftp
   ```

   The **stopsrc** command with the **-t** flag stops the **ftpd** subserver. The **stopsrc** command allows all pending connections to start and all existing connections to complete, but prevents new connections from starting. You must use the **-t** flag to specify a subserver. Otherwise, the command does not execute properly.
3. To force the **ftpd** daemon and all **ftpd** connections to stop, type the following:

   ```
   stopsrc -t -f ftp
   ```

   The **stopsrc** command with the **-t** and **-f** flags forces the **ftpd** subserver to stop. It terminates all pending connections and existing connections immediately.
4. To display a short status report about the **ftpd** daemon, type the following:

   ```
   lssrc -t ftp
   ```

   The **lssrc** command with the **-t** flag returns the daemon's name, process ID, and state (active or inactive). You must use the **-t** flag to specify a subserver. Otherwise, the command does not execute properly.

Files

/etc/locks/ftpd	Contains interlock and process ID (PID) storage.
/etc/group	Contains passwords for groups.
/etc/passwd	Contains passwords for users.

/etc/security/login.cfg	Contains configuration information for login and user authentication.
/etc/security/passwd	Contains encrypted passwords.
/etc/syslog.conf	Contains configuration information for the **syslogd** daemon.
/usr/samples/tcpip/anon.ftp	Contains the example shell script with which to set up an anonymous FTP account. This file also contains directions for its use.
/etc/ftpd.conf	Contains the configuration parameters for TLS support.

Related Information

The **ftp** command, **lssrc** command, **kill** command, **no** command, **rcp** command, **refresh** command, **rlogin** command, **rsh** command, **startsrc** command, **stopsrc** command, **telnet** command.

The **inetd** daemon, **syslogd** daemon.

The **kvalid_user** function.

The **/etc/ftpusers** file format, **/etc/inetd.conf** file format, **/etc/services**, $HOME/.k5login file format.

TCP/IP daemons in *Networks and communication management*.

Authentication and the secure rcmds in *Networks and communication management*.

Network option tunable parameters in *Performance management*.

---

fuser Command

Purpose

Identifies processes using a file or file structure.

Syntax

fuser [**-c** | **-d** | **-f**] [**-k** | **-K** { *SignalNumber* | *SignalName* }] [**-u**] [**-x**] [**-V**]*File* ...

Description

The **fuser** command lists the process numbers of local processes that use the local or remote files specified by the *File* parameter. For block special devices, the command lists the processes that use any file on that device.

Each process number is followed by a letter indicating how the process uses the file:

c	Uses the file as the current directory.
e	Uses the file as a program's executable object.
r	Uses the file as the root directory.
s	Uses the file as a shared library (or other loadable object).

The process numbers are written to standard output in a line with spaces between process numbers. A new line character is written to standard error after the last output for each file operand. All other output is written to standard error.

The **fuser** command will not detect processes that have mmap regions where that associated file descriptor has since been closed.

Flags

-c	Reports on any open files in the file system containing *File*.
-d	Implies the use of the **-c** and **-x** flags. Reports on any open files which have been unlinked from the file system (deleted from the parent directory). When used in conjunction with the **-V** flag, it also reports the inode number and size of the deleted file.
-f	Reports on open instances of *File* only.
-K *SignalNumber* | *SignalName*	Sends the specified signal to each local process. Only the root user can kill a process of another user. Signal can be specified as either a signal name, such as -9 or KILL for the SIGKILL signal. Valid values for *SignalName* are those which are displayed by the **kill -l** command.
-k	Sends the **SIGKILL** signal to each local process. Only the root user can kill a process of another user. **Note: fuser -k** or **-K** might not be able to detect and kill new processes that are created immediately after the program starts to run.
-u	Provides the login name for local processes in parentheses after the process number.
-V	Provides verbose output.
-x	Used in conjunction with **-c** or **-f**, reports on executable and loadable objects in addition to the standard fuser output.

Security

Attention RBAC users and Trusted AIX users: This command can perform privileged operations. Only privileged users can run privileged operations. For more information about authorizations and privileges, see Privileged Command Database in *Security*. For a list of privileges and the authorizations associated with this command, see the **/etc/security/privcmds** database file.

Examples

1. To list the process numbers of local processes using the **/etc/passwd** file, enter:
   ```
   fuser /etc/passwd
   ```
2. To list the process numbers and user login names of processes using the **/etc/filesystems** file, enter:
   ```
   fuser -u /etc/filesystems
   ```
3. To terminate all of the processes using a given file system, enter:
   ```
   fuser -k -x -u -c /dev/hd1
   ```
   or
   ```
   fuser -kxuc /home
   ```
   Either command lists the process number and user name, and then terminates each process that is using the **/dev/hd1 (/home)** file system. Only the root user can terminate processes that belong to another user. You might want to use this command if you are trying to unmount the **/dev/hd1** file system and a process that is accessing the **/dev/hd1** file system prevents this.
4. To list all processes that are using a file which has been deleted from a given file system, enter:
   ```
   fuser -d /usr
   ```

Files

/dev/kmem	Used for the system image.
/dev/mem	Also used for the system image.

Related Information

The **kill** command, **killall** command, **mount** command, and **ps** command.

For more information about the identification and authentication of users, discretionary access control, the trusted computing base, and auditing, refer to *Security*.

fwtmp Command

Purpose

Manipulates connect-time accounting records by reading binary records in **wtmp** format from standard input, converting them to formatted ASCII records. The ASCII version is useful when it is necessary to edit bad records.

Syntax

/usr/sbin/acct/fwtmp [**-i**] [**-c**] [**-X**]

Description

The **fwtmp** command manipulates the accounting records by reading binary records in **wtmp** format from standard input, converting them to formatted ASCII records.

Flags

-i	Accepts ASCII records in the **utmp** format as input.
-c	Converts output to **utmp** formatted binary records.
-ic	Converts ASCII **utmp** formatted input records to binary output records.
-X	Prints all available characters of each user name instead of truncating to the first 8 characters.

Security

Access Control: These commands should grant execute (x) access only to members of the **adm** group.

Examples

1. To convert a binary record in **wtmp** format to an ASCII record called `dummy.file`, enter:

   ```
   /usr/sbin/acct/fwtmp < /var/adm/wtmp > dummy.file
   ```

   The content of a binary **wtmp** file is redirected to a dummy ASCII file.
2. To convert an ASCII `dummy.file` to a binary file in **wtmp** format called `/var/adm/wtmp`, enter the `fwtmp` command with the `-ic` switch:

   ```
   /usr/sbin/acct/fwtmp  -ic < dummy.file  > /var/adm/wtmp
   ```

   The dummy ASCII file is redirected to a binary **wtmp** file.

Files

/usr/sbin/acct/fwtmp	Contains the **fwtmp** command.
/var/adm/wtmp	Contains records of date changes that include an old date and a new date.
/usr/include/utmp.h	Contains history records that include a reason, date, and time.

Related Information

The **acctcon1** or **acctcon2** command, **acctmerg** command, **acctwtmp** command, **runacct** command, **wtmpfix** command.

Setting up an accounting system in *Operating system and device management* describes the steps you must take to establish an accounting system.

See the Accounting commands in *Operating system and device management* for a list of accounting commands that can be run automatically or entered from the keyboard and about the preparation of daily and monthly reports, and the accounting files.

fxfer Command

Purpose

Transfers files between a local system and a host computer connected by HCON.

Syntax

To Restart an Interrupted File Transfer

fxfer -R [**-n** *SessionName*]

To Download a File from the Host

fxfer [**-n** *SessionName*] [**-a** | **-r**] [**-d**] [**-c** | **-C**] [**-J**] [**-f** *FileName*] [**-F**] [**-H** *HostType*][**-l** *InputField*] [**-q**] [**-t** [[**-l**] [**-s**] [**-b**]] | **-T** [[**-l**] [**-s**] [**-b**]]]

[**-v**] [**-x** *HostLogin*] [**-e**] [**-X** *CodeSet*] *SourceFile DestFile*

To Upload a File to the Host

fxfer [**-n** *SessionName*] [**-a** | **-r**] [**-u**] [**-c** | **-C**] [**-J**] [**-f** *FileName*] [**-H** *HostType*] [**-q**] [**-t** [[**-l**] [**-s**]] | **-T** [[**-l**] [**-s**]]] [**-l**] [**-s**] [**-v**] [**-x** *HostLogin*] [**-X** *CodeSet*] [**-F** | **-V** | **-U**] [**-B** *BlockSize*] [**-L** *LoglRecLength*] [**-l** *InputField*] [**-S** *NumberUnits* [**,***IncreaseUnits* | **,***IncreaseUnits***,***UnitType* | **,,***UnitType*]] [**-M** *Volume*] [**-N** *Unit*] [**-k**] *SourceFile DestFile*

To Display the Help Screen

fxfer -h

Description

The **fxfer** command transfers files between local system and mainframe hosts connected by the Host Connection Program (HCON). Files may transfer from a local system to the host (uploading) or from the host to a local system (downloading). The **fxfer** command transfers the file named by the *SourceFile* parameter to the file named by the *DestFile* parameter. The transfer occurs over an HCON session requiring a specific session profile or an existing session.

The host operating system may be VM/CMS, MVS/TSO, CICS/VS (for CICS/MVS or CICS/VSE), VSE/ESA, or VSE/SP, with the corresponding version of the 3270 File Transfer Program (**IND$FILE** or its equivalent) installed. The version of the host file transfer program is determined by the File Transfer Program value in the session profile. The **fxfer** command supports transfer of either text or binary data. Files will transfer to or from the host with or without ASCII or EBCDIC translation.

Security mechanisms prevent unauthorized access, the destruction of existing files, or the loss of data. If a non-HCON user issues the **fxfer** command, the command fails. If the **fxfer** command is interrupted before completion, the state of the transfer is saved in a RESTART file.

If the **fxfer** command is issued with the **-h** flag, it displays a help screen. If the command is issued with the **-R** flag, it searches the **$HOME** directory for a restart file. If a restart file exists, the restart menu displays, enabling a restart of the file transfer. If the **-h** and **-R** flags are not specified, the command attempts to perform the specified file transfer.

The **fxfer** command information includes:

- Flags
- Flags for Host File Characteristics
- Examples
- Files

This command requires:

- One or more adapters used to connect to a mainframe host.
- One of the following mainframe operating systems be installed on the host:
  - VM/SP CMS
  - VM/XA CMS
  - MVS/SP™ TSO/E
  - MVS/XA™ TSO/E
  - CICS/VS (for CICS/MVS or CICS/VSE)
  - VSE/ESA
- The mainframe Host-Supported File Transfer Program (**IND$FILE** or equivalent) be installed on the mainframe.

Session Profiles for Using the fxfer Command

The **fxfer** command communicates with an HCON session and may require a specific session profile. The session profile defines:

- Communication path to the host
- Host type
- Default file transfer direction (down or up)
- Recovery time
- File transfer wait period

When the **fxfer** command is performing an automatic logon, the profile can also define:

- Host logon ID
- AUTOLOG node ID
- Whether the AUTOLOG trace is on
- AUTOLOG time out value

The user usually specifies a session profile when invoking the **fxfer** command. The exception occurs when the command is run from a subshell of an existing session. In this case, if the user does not specify a session profile, the **fxfer** command uses the existing session. If the appropriate session is not running, the **fxfer** command attempts to invoke a new session.

The **fxfer** command searches for an HCON session as follows:

- When issued without the **-n** *SessionName* flag:
  - If the **fxfer** command is issued from a subshell of an existing session, the command uses the session associated with the subshell (defined by the **$SNAME** environment variable).
  - If *not* issued from a subshell of an emulator session, the **fxfer** command issues an error message and terminates.
- When issued with the **-n** *SessionName* flag, the file transfer performs over the specified session. If the specified session does not exist, the command searches for a session profile for that session. If the specified session profile cannot be found, the **fxfer** command issues an error message and terminates. If the specified profile exists, the **fxfer** command attempts an automatic logon to the host using either the AUTOLOG values defined in the session profile, the values defined with the **-x** flag, or by prompting the user for the necessary logon information.

Interrupted and Restarted File Transfers

The **fxfer** command can be interrupted by the operator or an unrecoverable communication error, before completion. If interrupted, the command saves the state of the transfer in a RESTART file. The transfer can be restarted from the beginning without loss of data.

If you run a new file transfer after an interrupted transfer, the **fxfer** command signals that a RESTART file has been created and displays these choices:

- Restart the interrupted file transfer.
- Save the RESTART file and exit the file transfer program.
- Delete the RESTART file and exit the file transfer program.
- Delete the RESTART file and continue the present transfer.

The **fxfer** command with the **-R** flag also restarts an interrupted file transfer.

If the host communication is lost or disconnected during a file transfer started with an automatic logon, the file transfer attempts to recover by reconnecting and logging back on to the host. The recovery time for this attempt is determined by the File Transfer Recovery Time value in the session profile. Once the host connection is re-established, the file transfer resumes from the start. If communication cannot be re-established, the file transfer program generates a RESTART file.

When an explicit file transfer loses communication with the host, the user must restart the emulator session and log back in to the host before attempting to restart the file transfer.

Source and Destination Files

The **fxfer** command *SourceFile* and *DestFile* parameters are required. The *SourceFile* parameter specifies the source file for a file transfer. The *DestFile* parameter specifies the destination file for a file transfer. The local system file names are in the normal format. The host file names conform to the host naming convention, which is one of the following formats:

Host Type **File Name Format**

VM/CMS *"FileName FileType FileMode"*

Note: The " " (double quotation marks) are required for all VM/CMS file names to ensure proper file transfer.

MVS/TSO *"['DataSetName* [**(***MemberName***)**] [**/***Password*]['] *"*

where:

DataSetName
Indicates either a physical sequential data set or a partitioned data set.

(*MemberName***)**
Indicates the name of one of the members in the directory of an existing partitioned data set. The **()** (parentheses) enclosing the *MemberName* are required.

/*Password*
Required if password protection is specified for the MVS/TSO data set. The **/** (slash) preceding the *Password* is required.

Notes:

1. The " " (double quotation marks) are required for all MVS/TSO file names to ensure proper file transfer.
2. When specifying a complete path name for MVS/TSO file names, use ' (single quotation marks) within the " (double quotation marks). Do not put spaces between the double and single quotation marks or between the quotation marks and the file names.

CICS/VS *"FileName"*

Host Type	File Name Format
VSE/ESA	*"FileName FileType"*

Notes:

1. The " " (double quotation marks) are required for all CICS/VS, VSE/ESA, and VSE/SP file names to ensure proper file transfer.
2. CICS/VS, VSE/ESA, and VSE/SP file name conventions allow for a file name up to 8 characters long.
3. In a DBCS environment, HCON does not support a VSE host.

Flags

Note: For Double-Byte Character Set (DBCS) support that includes either Japanese-English, Japanese Katakana, Korean, or Traditional Chinese, these considerations apply:

- If the DBCS **-I** or **-s** flag is specified, one of the translate flags (**-t**, **-T**, or **-J**) must also be specified or the DBCS flags are ignored.
- The **-M**, **-N**, and **-k** flags are used only with MVS/TSO hosts.
- The **-e** flag is valid only with CICS for downloading.
- The **-b** flag is valid only for downloading.

-a Appends the file designated by *SourceFile* to the file designated by *DestFile*, if the destination file exists. This flag is ignored and the destination file is created if the file designated by *DestFile* does not exist.

> **Note:** The **-a** flag is not valid when uploading a file to a CICS/VS host. For VSE/ESA, the **-a** flag is valid only for uploading to CICS temporary storage (FILE=TS).

-b Retains the blanks at the end of each record when used with the **-t**, **-T**, **-c**, or **-C** flags. The **-b** flag is only supported in the DBCS environment.

-c In a DBCS environment, the **-c** flag changes LF (line-feed) code of a file to CRLF (carriage return line-feed) code if the file transfer is an upload. For a downloading file transfer, the **-c** flag changes the CRLF code of a file to LF code.

-C In a DBCS environment, the **-C** flag inhibits the sending of the EOF (end-of-file) code of a PC-DOS file if the file transfer is an upload. For a downloading file transfer, the **-C** flag appends an EOF code: `x'1A` at the end of a PC-DOS file.

-d Downloads the file by transferring it from the host to the local system. If neither this flag nor the **-u** flag is specified, the File Transfer Direction characteristic in the session profile determines the direction of the transfer.

> **Note:** When downloading a translated file from a VSE/ESA host file transfer (FILE=HTF) the file is deleted from the host system unless you specify the **-I** `"KEEP"` flag.

-e Deletes the temporary storage queue at the completion of the file transfer. Use this flag only with the CICS host for downloading. The **-e** flag is only supported in the DBCS environment.

-f *FileName* Places the file transfer process diagnostic output (or file transfer status) in the file specified by the *FileName* variable.

If the **-f** flag is not specified for an asynchronous transfer, messages are placed in the **$HOME/hconerrors** file. If the **-f** flag is not specified for a synchronous transfer, messages are sent to standard output.

Messages due to errors in specifying file transfer parameters or file names, or failures in the file transfer process, are directed to standard output (if it is a local system screen) or to the **$HOME/hconerrors** file (if standard output is not a local system screen).

-h Displays a help screen for the **fxfer** command. This screen summarizes each available command flag and command operation. When this flag is specified all other flags are ignored and no files are transferred.

Notes:

1. If the **-h** flag is used, all other flags are ignored. No files transfer.
2. If the **fxfer** command is not initiated from a subshell of an existing HCON session, either the **-h** flag or the **-n** flag is required.

-H *HostType* Specifies the type of host. The *HostType* variable may have any of these values:

CMS VM/SP CMS or VM/XA CMS

TSO MVS/SP TSO or MVS/XA TSO

CICS CICS/VS (The CICS host type includes CICS/VSE, CICS/MVS, CICS/ESA®, and CICS/MVS/ESA.)

VSE VSE/ESA (Not supported in a DBCS environment.)

If the **-H** flag is omitted, the value specified by the Host Type characteristic in the session profile is used. The user must specify the correct host operating system.

Notes:

1. If you specified the **CICS** or **VSE** value and the system returns an error, retry the command with the alternate value. The CICS and VSE **IND$FILE** programs are functionally interchangeable; however, there is a 6-byte header-size discrepancy that makes the versions operationally incompatible. The destination host may be using the alternate version of the program.
2. To transfer files to an MVS/TSO host, you may need to leave session manager mode before initiating the file transfer.

-I *InputField* Specifies host file transfer options placed directly within the **IND$FILE** command. Also allows comments within the **IND$FILE** command placed after a) (right parentheses). The value specified by the *InputField* variable is placed in quotation marks, as follows:

```
-I   "FILE=TS)   This   is   a   comment"
```

Note: The **-I** field is not supported in a DBCS environment.

-J Allows data conversion between EBCDIC and ASCII, and normalization of SI/SO characters. The translation depends on the direction of the transfer:

Upload
Translates 1-byte characters of a file to EBCDIC code. For DBCS countries, the extended code is translated to the appropriate DBCS code. SO/SI characters are inserted into DBCS fields containing DBCS characters. If the file contains control codes 0x1E or 0x1F, they are replaced with SO and SI characters respectively.

Download
Translates EBCDIC code to 1-byte characters of a file; For DBCD, the DBCS code is translated to extended code. Deletes SO/SI characters from DBCS fields.

Note: The **-J** field is only supported in a DBCS environment.

-k Releases unused records in the dataset at the completion of file transfer. Use this flag only in the MVS/TSO environment. The **-k** flag is only supported in the DBCS environment.

-I Specifies the host language in the DBCS environment. This option must be used with one of the translate flags (**-t**, **-T**, or **-J**). If **-t**, **-T**, or **-J** is omitted, the **-I** flag is ignored. If the **-I** flag is not specified, the host language defined in the session profile is used. If the **-I** flag is specified, the host language used is the alternate language of the language defined in the session profile. For example, if the Language characteristic in the session profile is JPK (Japanese Katakana), the host language used for file transfer will be Japanese-English. The **-I** flag is only supported in the DBCS environment.

-M *Volume* Specifies the volume serial number of the host disk for dataset allocation. Use this flag only in the MVS/TSO environment. The **-M** flag is only supported in the DBCS environment.

-n *SessionName* Specifies the name of a previously defined session whose characteristics control the file transfer. The session name is a single character in the range of a to z. Capital letters are interpreted as lowercase letters.

The **-n** *SessionName* flag is required except when the user is initiating the **fxfer** command from a subshell of an existing session. In this case, if the **-n** flag is not used the **fxfer** command defaults to the existing session.

Notes:

1. The specified session must have been previously defined using the Web-based System Manager, the **smit hcon** fast path command or the **mkhcons** command.
2. If the **fxfer** command is not initiated from a subshell of an existing HCON session, either the **-h** flag or the **-n** flag is required.

-N *Unit* Specifies the unit type of the host disk for dataset allocation. Use this flag only in the MVS/TSO environment. The **-N** flag is only supported in the DBCS environment.

-q Runs the file transfer asynchronously as a background process. If any file transfers are not completed, the current transfer request is queued. If the **-q** flag is not specified, the file transfer operation is synchronous. If the **-f** flag is not specified, diagnostic output and status is placed in the **$HOME/hconerrors** file.

Note: The system limits the number of bytes allowed in one Interprocess Communication (IPC) message queue. As a result, the maximum number of file transfers that can be queued at any one time is approximately 580.

-r Specifies replacement of an existing file on the host (upload) or an existing file on the local system (download). On downloads, the replacement is done only when the transfer is successful. This ensures the existing file is not lost or destroyed if the transfer does not complete for any reason.

If the **-r** flag is specified and the file does not exist, it is created during the file transfer. If the **-r** flag is *not* specified and the destination file exists, an error message is produced.

For uploading, the **-r** flag must be specified when using a version of the host file transfer program below PTF UR20455 for MVS/TSO or PTF UR90118 for VM/CMS. For VSE and CICS the **-r** flag is ignored.

Note: The host file transfer program usually defaults to replace for a file. If it does not, add `-I "replace"` to the **fxfer** command to specify replace.

Attention: When replacing a file on the host, you must specify a logical record length (**-L** flag) and a record format (**-F** or **-V** flag) equal to the logical record length and record format of the existing file. If you do not do this, data corruption may result. This does not apply to VSE/ESA.

-R Restarts a previous file transfer (which was interrupted by the user or an unsuccessful recovery attempt) using the information saved in one of the RESTART files: the **$HOME/x_fxfer.r** file or the **$HOME/i_fxfer.r** file. If the file transfer is not invoked from the subshell of an existing session, the **-n** *SessionName* flag must be included to specify the session to be used. If the **-R** flag is specified in conjunction with any other file transfer flags, those flags are ignored and the RESTART file transfer menu is displayed.

> **Note:** With the **-R** flag, all other flags except the **-n** *SessionName* flag are ignored. The RESTART file transfer menu displays.

-s Specifies the SO/SI handling in the DBCS environment. The **-s** flag must be used with one of the translate flags (**-t**, **-T**, or **-J**). If **-t**, **-T**, or **-J** is omitted, the **-s** flag is ignored. When the **-s** flag is specified, the following functions are performed for file transfer:

Upload
SO/SI characters are not inserted in DBCS fields.

Download
SO/SI characters are replaced with control characters (0x1E/0x1F) in DBCS fields.

The **-s** flag is only supported in the DBCS environment.

-t Performs ASCII-EBCDIC translation for a file. If downloading, the **fxfer** command translates EBCDIC to ASCII. If uploading, the **fxfer** command translates ASCII to EBCDIC. The language is specified by the Language characteristic in the session profile. The **-t** flag assumes the file is a text file. The new-line character is the line delimiter.

When the **-t** flag is used in a DBCS environment with other DBCS supported flags, the behavior of the **-t** flag changes as follows:

Upload
Translates JISCII (Japan) or ASCII (Korean, Traditional Chinese) to EBCDIC. Inserts SO/SI characters in DBCS fields.

Download
Translates EBCDIC to JISCII (Japan) or ASCII (Korean, Traditional Chinese). Deletes SO/SI characters from DBCS fields.

-T Performs ASCII-EBCDIC translation for a disk operating system file. The character sequence, CRLF, used as the line delimiter, and a disk operating system EOF (end-of-file) character are inserted at the end of the downloaded file. The language to be used for EBCDIC to ASCII translation is specified by the Language characteristic in the session profile. The **-T** flag is used to translate disk operating system files.

> **Note:** If neither the **-T**, **-t**, nor the **-J** flag is specified, the file transfer assumes no translation and transfers the information in binary form.

-u Uploads the file by transferring the file from the local system to the host. If neither this flag nor the **-d** flag is specified, the File Transfer Direction characteristic in the session profile determines the direction of the transfer.

-v Periodically writes the current status of the file transfer to the screen or to the status file specified by the **-f** flag. The status includes the number of bytes transferred and the elapsed time since the file transfer process began transferring data.

-x *HostLogin*

Uses the login ID specified by the *HostLogin* variable to log in to the host. The user is prompted to enter the password.

The *HostLogin* string consists of the host login ID, the AUTOLOG node ID, and other optional AUTOLOG values. The string cannot contain any blanks and must contain the AUTOLOG node ID. Format the AUTOLOG string as:

```
UserID,AutologNodeID[,Trace,Time   .   .   .]
```

If the **-x** flag is not specified, the information for the *HostLogin* string is taken from the session profile as follows:

- If the host login ID is set in the session profile, you are prompted for the password. The remaining parameters are retrieved from the profile.
- If the host login ID is not set in the profile, you are prompted for both the host login string and the password.
- Your response to a prompt always overrides a profile parameter. For example, if the AUTOLOG time is set in the profile but you enter a different value at the prompt, the value entered at the prompt is used.

If you omit certain parameters from the host login string, they are retrieved from the profile, if defined there. For example, if the you set the AUTOLOG Node ID, AUTOLOG Trace, and AUTOLOG Time parameters in the profile, only the host login ID must be entered at the prompt.

The file transfer process logs in to the host and establishes an emulation session using the session profile specified with the **-n** flag. Once the process is successfully logged in, the file transfer begins.

The File Transfer Wait Period parameter in the session profile determines how long the login session is maintained. Using this parameter, the host login session is maintained for subsequent file transfers. The need to log in again is eliminated.

-X *CodeSet*

Specifies an alternate code set to use for ASCII-EBCDIC translation. If the **-X** flag is omitted, the code set specified by the system locale is used. The following code sets are supported:

Default
: Uses current system ASCII code page.

IBM-932
: Uses IBM code page 932 for translation in a DBCS environment.

ISO8859-1
: Uses ISO 8859-1 Latin alphabet number 1 code page.

ISO8859-7
: Uses ISO 8859-7 Greek alphabet.

ISO8859-9
: Uses ISO 8859-9 Turkish alphabet.

IBM-eucJP
: Uses IBM Extended UNIX Code for translation in the Japanese Language environment.

IBM-eucKR
: Uses IBM Extended UNIX Code for translation in Korean Language environment.

IBM-eucTW
: UsesIBM Extended UNIX Code for translation in Traditional Chinese Language environment.

Flags for Host File Characteristics

The following flags specify host file characteristics and can be used only to upload files (with the exception of the **-F** flag, which can be used when downloading from a VSE host):

-B *BlockSize* Specifies the block size of the host data set. The **-B** flag can only be used in the MVS/TSO environment and only for sequential data sets. The *BlockSize* variable cannot exceed the capacity of a single track. The **-B** flag is ignored if the file is being appended. A block size value of 0 causes an error.

-F Specifies fixed-length records. This is the default if neither the **-V**, **-t**, **-T**, **-c**, nor **-C** flag is specified. The **-F** flag is ignored if the file is being appended.

On a CICS or VSE host, one of the translate flags (**-t** or **-T**) or one of the CRLF flags (**-c** or **-C**) must be specified along with the **-F** flag, since the CICS and VSE host file transfer programs do not support fixed record lengths. The combination of the **-F** flag and the translate flag causes the transfer program to pad the records with blanks to the end of the logical record length. The default is 80.

> **Note:** Use the **-F** flag when downloading from a VSE host to prevent the deletion of trailing blanks from the translated file.

-L *LoglRecLength* Specifies the logical record length in bytes of the host file. For new files, the default is 80. For variable-length records, *LoglRecLength* is the maximum size of the record. The **-L** flag is ignored if the file is being appended. A *LoglRecLength* value of 0 causes an error.

Because of MVS overhead, the actual number of bytes stored in the variable length records on an MVS/TSO host is four bytes less than the value specified by the *LoglRecLength* variable.

The CICS and VSE host file transfer programs do not support logical record lengths. For transfers to or from a CICS or VSE host the **-L** flag must be accompanied by the **-F** flag. The combination of the **-F** and **-L** flags causes the transfer program to pad the records with blanks to the end of the logical record length. The default is 80.

> **Note:** The **-L** flag is required if a record length is greater than the default record length of 80.

-S *NumberUnits* [**,***IncreaseUnits* | **,***IncreaseUnits***,***UnitType* | **,,***UnitType*]

Specifies the amount of space to be allocated for a new sequential data set on TSO. For large MVS files, the maximum block size permissible on the host is used to ensure that the whole disk track is filled. The **-S** flag can be used only with MVS/TSO hosts.

The following variables can be used with the **-S** flag. If used, they must be specified in the order given and separated by commas. If a variable preceding another variable is omitted, a comma must be included as a placeholder. A space is required between the **-S** flag and the *NumberUnits* variable. However, no spaces can appear in the variable string.

NumberUnits
: Specifies the number of units of space to be added initially. A value of 0 or a negative value cannot be specified for the *NumberUnits* variable.

IncreaseUnits
: Specifies the number of units of space to be added to the data set each time the previously allocated space is filled (optional).

UnitType
: Defines the unit of space and may be `T` for tracks, `C` for cylinders, or a number specifying the average block size (in bytes) of the records written to the data set. If the *UnitType* variable is not specified, the default is the value specified by the **-B** flag. If the **-B** *BlockSize* flag is not specified, the default value is 80.

Following are the possible combinations of variables used with the **-S** flag:

-S *NumberUnits***,***IncreaseUnits***,***UnitType*

-S *NumberUnits***,***IncreaseUnits*

-S *NumberUnits*

-S *NumberUnits***,,***UnitType*

-U Specifies records of undefined length. The **-U** flag can only be used in the MVS/TSO environment. The **-U** flag is ignored if the file is being appended.

-V Specifies records of variable length. This is the default if the **-F** flag is not specified, and either the **-t**, **-T**, **-c**, or **-C** flag is specified. The **-V** flag is ignored if the file is being appended.

The **-V** flag is not supported by the CICS or VSE host file transfer programs, since variable record lengths are the default.

Examples

The following examples assume the session profile for session `a` is:

```
Session   type         DFT
Communication   device         3270c0
Language         English   (U.S.A.)
Host   type         CMS
File   transfer   direction         up
File   transfer   wait   period         10
File   transfer   recovery   time         30
```

where:

- The host type is VM/CMS.
- The connection is made using the DFT 3270 connection device.
- The file transfer default direction is upload (to use session profile `a` for downloading files, the user must specify the **-d** flag with the **fxfer** command).
- The file transfer process stays logged in for 10 minutes.
- If a transfer is interrupted, the process attempts recovery for 30 minutes before saving information in the RESTART file for later transfer.
- The translation language is U.S.A. ASCII-EBCDIC.

1. To upload the `samplefile` file (in the current directory) to the host and translate it to EBCDIC using the U.S.A. translation table, enter:

   ```
   fxfer   -n   a   -t   samplefile   "test   file   a"
   ```

   – **-n** instructs the **fxfer** command to use session `a` to transfer the file.

   – **-t** instructs the command to translate using the new-line character.

   The translated data is placed in the `test file a` on the host. Because the host file name contains spaces, quotation marks around the file name are required.

2. To upload the `file2` file to the VM/CMS host `test file b`, enter:

   ```
   fxfer   -urv   -L   132   -V   -H   CMS   file2   "test   file   b"
   ```

   – **-u** instructs the **fxfer** command to upload the file.

   – **-H** indicates that the host type is a VM/CMS host. If the destination file exists, it is replaced (since the **-r** flag is specified) by the transferred file.

   – **-v** causes **fxfer** to display the number of bytes transferred and elapsed time. The status or diagnostic output is displayed on the terminal.

   – If the host file does not exist, the host file maximum logical record length is set to 132 bytes (**-L** flag).

   – The host file record format is variable (**-V** flag). No translation is performed.

3. To upload, from a subshell of emulator session `a`, the local system **/etc/motd** file to the CICS motdfile host file with translation and padding of blanks, enter:

   ```
   fxfer   -utFH   CICS   -I   ")This   is   a   comment"   /etc/motd   "motdfile"
   ```

   – **-u** instructs the command to upload the file.

   – **-t** causes translation from ASCII to EBCDIC.

   – **-F** causes the transfer program to pad the uploaded file with blanks to column 80 (the default record length). To change the default column, use the **-L** flag with a different record length (column).

- **-H** specifies the host as type CICS.
- **-I** specifies that the *InputField* value be added to the **IND$FILE** command.

In this example, "This is a comment" is a host comment field.

To upload or download files with the **fxfer** command, to or from a TSO environment other than your current environment, you must have authorization for the other environment. You must completely qualify the file (or dataset) within single quotes ('), then double quotes (" ").

4. For example, to upload the file newfile to a TSO environment where the complete qualified name is sys4.parmlib.samplefile, enter:

```
fxfer   -urtvH   TSO   'newfile'   "sys4.parmlib.samplefile"
```

   - **-u** instructs the command to upload the file.
   - If the sys4.parmlib.samplefile file exists, it is replaced (**-r** flag) with the translated contents of the newfile file (**-t** flag).
   - **-v** instructs the **fxfer** command to write the file transfer status to the local screen every few seconds.
   - **-H** instructs the **fxfer** command that the host is a MVS/TSO host.

     Note: This example assumes that the **fxfer** command is issued from a subshell of an established session (use the **e789** command to establish a session).

5. To download the file spfuser.test from the MVS/TSO host to the local system, enter:

```
fxfer   -n   a   -d   -r   -H   TSO   spfuser.test   samplefile1
```

   - **-n** instructs the **fxfer** command to use session a to transfer the file. If session a has not already been established, the command attempts an automatic login. Since no host login ID is specified, the **fxfer** command checks the session profile for a login ID. If one is not specified there, the user is prompted for the login ID and password.
   - **-d** overrides the default file transfer direction of upload.
   - If the samplefile1 file already exists, it is replaced (**-r** flag) with the downloaded file from the host.
   - **-H** instructs the **fxfer** command that the host is an MVS/TSO host instead of VM/CMS (the default from the session profile).

   The transferred file is placed in the samplefile1 file on the local system. The file transfer is performed synchronously.

6. To download the VM/CMS host test file a and append it to the local system mydir/samplefile file, using session profile a and automatic login, enter:

```
fxfer   -n   a   -dat   -q   -f   status.out
-x   laura,vm1,trace   "test   file   a"   mydir/samplefile
```

   - **-n** instructs the **fxfer** command to use session profile a to transfer the file.
   - **-x** provides the host login ID. The **fxfer** command first checks to see if session is established on the local system. If so, the command transfers the file over the existing session. If session a is not established, the **fxfer** command performs an automatic login using the host logon ID laura and the AUTOLOG script vm1, and traces the login activity. The user is prompted for the password. The command transfers the file.
   - **-dat** instruct the **fxfer** command to download the file (**-d** flag), translate the data from EBCDIC to ASCII (**-t** flag) using the U.S.A. translation table (defined in the session profile), and append (**-a** flag) the translated file to the mydir/samplefile file on the local system. If the mydir/samplefile file does not already exist, the **fxfer** command ignores the **-a** flag and creates the file.
   - The status or diagnostic output is placed in the status.out file in the current local directory (**-f** flag).
   - **-q** instructs the **fxfer** command to transfer the file asynchronously.

   When the user enters the password, the prompt is returned and the file transfer is performed in the background.

To queue another file transfer to be performed by the same file transfer process, enter:

```
fxfer -n a -daq -f status.out "test file b"
mydir/samplefile
```

– **-n** instructs the **fxfer** command to use session `a` to transfer the file. Since session `a` has been established by the previous command, the **fxfer** command does not need to log in to the host again.
– **-d** instructs the command to download a file from the host.
– **-a** instructs the command to append the `test file b` host file to the `mydir/samplefile` file on the local system.
– **-q** instructs the **fxfer** command to transfer the file asynchronously.

The **fxfer** command continues to send status information to the `status.out` file on the local system (**-f** flag).

Notes:

a. If the text for the **fxfer** command extends beyond the limit of the screen, the text wraps automatically to the next line. Pressing the Enter key to wrap the text causes an error.
b. Attempting to start a synchronous file transfer when there is an asynchronous transfer in the queue causes an error.
c. The user will not be prompted for a login ID or a password as long as the session remains running and the **dfxfer** process remains logged in to the host. The amount of time the process remains logged in is determined by the File Transfer Wait Period in the session profile.

7. To restart an interrupted file transfer from an emulator subshell, enter:

```
fxfer -R
```

-R instructs the **fxfer** command to use the information saved in one of the RESTART files to execute a file transfer. The RESTART file is the **$HOME/x_fxfer.r** explicit restart file or **$HOME/i_fxfer.r** implicit restart file. If the **-R** flag is specified in conjunction with other file transfer flags, the other flags are ignored. The RESTART file transfer menu is displayed. Using this menu, instruct the **fxfer** command to transfer the interrupted file.

8. To restart the file transfer from the command line instead of from an emulator subshell, enter:

```
fxfer -R -n a
```

The **-n** flag instructs the **fxfer** command to use session `a` to perform the restarted transfer.

Files

/usr/bin/fxfer	Contains the **fxfer** command.
/usr/bin/dfxfer	Contains the **dfxfer** process.
$HOME/i_fxfer.r	Contains RESTART information for automatic login queues. Temporary file created by the **fxfer** command.
$HOME/x_fxfer.r	Contains RESTART information for manual login queues. Temporary file created by the **fxfer** command.
$HOME/hconerrors	Contains HCON diagnostic output and file transfer status. Temporary file created by any HCON command.
/usr/lib/libfxfer.a	Contains the library for programmatic file transfers.

Related Information

smit command.

For information on installing the Web-based System Manager, see Chapter 2: Installation and System Requirements in *AIX Version 6.1 Web-based System Manager Administration Guide*.

gated Daemon

Purpose

Provides gateway routing functions for the RIP, RIPng, EGP, BGP, BGP4+, HELLO, IS-IS, ICMP, ICMPv6, and SNMP protocols.

> **Note:** Use SRC commands to control the **gated** daemon from the command line. Use the **rc.tcpip** file to start the daemon with each system startup.

Syntax

/usr/sbin/gated [**-c**] [**-C**] [**-n**] [**-N**] [**-t** *TraceOptions*] [**-f** *ConfigFile*] [*TraceFile*]

Description

The **/usr/sbin/gated** daemon handles multiple routing protocols and replaces **routed** and any routing daemon that speaks the (HELLO) routing protocol. The **/usr/sbin/gated** daemon currently handles the Routing Information Protocol (RIP), Routing Information Protocol Next Generation (RIPng), Exterior Gateway Protocol (EGP), Border Gateway Protocol (BGP) and BGP4+, Defense Communications Network Local-Network Protocol (HELLO), and Open Shortest Path First (OSPF), Intermediate System to Intermediate System (IS-IS), and Internet Control Message Protocol (ICMP)/Router Discovery routing protocols. In addition, the **gated** daemon supports the Simple Network Management Protocol (SNMP). The **gated** process can be configured to perform all of these protocols or any combination of them. The default configuration file for the **gated** daemon is the **/etc/gated.conf** file. The **gated** daemon stores its process ID in the **/etc/gated.pid** file.

> **Note:** Unpredictable results may occur when the **gated** and **routed** daemons are run together on the same host.

If on the command line a trace file is specified, or no trace flags are specified, the **gated** daemon detaches from the terminal and runs in the background. If trace flags are specified without specifying a trace file, **gated** assumes that tracing is desired to **stderr** and remains in the foreground.

Note: IS-IS routing protocol cannot be run on 64-bit kernel.

Signals

The **gated** server performs the following actions when you use the **kill** command to send it signals.

SIGHUP Re-read configuration.

A **SIGHUP** causes gated to reread the configuration file. The **gated** daemon first performs a clean-up of all allocated policy structures. All BGP and EGP peers are flagged for deletion and the configuration file is reparsed.

If the reparse is successful, any BGP and EGP peers that are no longer in the configuration are shut down, and new peers are started. The **gated** daemon attempts to determine if changes to existing peers require a shutdown and restart.

> **Note:** Reconfiguration is disable when OSPF (Open Shortest Path First) is enabled.

SIGINT Snapshot of current state.

The current state of all gated tasks, timers, protocols and tables are written to **/var/tmp/gated_dump**.

This is done by forking a subprocess to dump the table information so as not to impact the **gated** daemon's routing functions.

SIGTERM Graceful shutdown.

Upon receiving a **SIGTERM** signal, the **gated** daemon attempts a graceful shutdown. All tasks and protocols are asked to shutdown. Most will terminate immediately, the exception being EGP peers which wait for confirmation. It may be necessary to repeat the **SIGTERM** once or twice if this process takes too long.

All protocol routes are removed from the kernel's routing table on receipt of a **SIGTERM**. Interface routes, routes with **RTF_STATIC** set (from the **route** command where supported) and static routes specifying **retain** will remain. To terminate the **gated** daemon with the exterior routes intact, use the **SIGKILL** or **SIGQUIT** signals (which causes a core dump).

SIGUSR1 Toggle tracing.

Upon receiving a **SIGUSR1** signal, the **gated** daemon will close the trace file. A subsequent **SIGUSR1** will cause it to be reopened. This will allow the file to be moved regularly.

> **Note:** It is not possible to use the **SIGUSR1** signal if a trace file has not been specified, or tracing is being performed to **stderr**.

SIGUSR2 Check for interface changes.

Upon receiving a **SIGUSR2** signal, the **gated** daemon rescans the kernel interface list looking for changes.

The gated and snmpd Daemons

The **gated** daemon is internally configured to be an SNMP multiplexing (SMUX) protocol peer, or proxy agent, of the **snmpd** daemon. For more information, refer to ″SNMP daemon processing″ in *Networks and communication management*.

Manipulating the gated Daemon with the System Resource Controller

The **gated** daemon can be controlled by the System Resource Controller (SRC). The **gated** daemon is a member of the SRC **tcpip** system group. This daemon is disabled by default and can be manipulated by the following SRC commands:

startsrc	Starts a subsystem, group of subsystems, or a subserver.
stopsrc	Stops a subsystem, group of subsystems, or a subserver.
refresh	Causes the subsystem or group of subsystems to reread the appropriate configuration file.
lssrc	Gets the status of a subsystem, group of subsystems, or a subserver.

> **Note:** On initial startup from the **startsrc** command, the **gated** daemon does not start responding to other SRC commands until all **gated** initialization is completed. A very large **/etc/gated.conf** file can require a minute or more to parse completely.

Flags

-c	Specifies parsing of the configuration file for syntax errors after which the **gated** daemon exits. If no errors occur, the **gated** daemon puts a dump file into the **/var/tmp/gated_dump** file. The **-c** flag implies the **-tgeneral,kernel,nostamp** flag. If the **-c** flag is specified, the **gated** daemon ignores all **traceoption** and **tracefile** clauses in the configuration file.
-C	Specifies that the configuration file is parsed only for syntax errors. The **gated** daemon exists with a status of **1** if it finds any errors and with a status of **0** if it does not. The **-C** flag implies the **-tnostamp** flag.
-f *ConfigFile*	Specifies an alternate configuration file. By default, the **gated** daemon uses the **/etc/gated.conf** file.
-n	Specifies that the **gated** daemon will not modify the kernel's routing table. This is used for testing **gated** configurations with actual routing data.

-N Specifies that the **gated** daemon does not daemonize. Normally, if tracing to **stderr** is not specified and the parent process ID is not 1, the **gated** daemon daemonizes. This flag allows the use of a method similar to **/etc/inittab** of invoking the **gated** daemon that does not have a process ID of 1.

-t*TraceOptions* Specifies which trace options are enabled at system startup. When used without the *TraceOptions* variable, this flag starts the **general** trace options. Separate each trace option from another with a comma. Do not insert a space between the flag and the first trace option.

The **-t** flag must be used to trace events that take place before the **/etc/gated.conf** file is parsed, such as determining the interface configuration and reading routes from the kernel.

The **gated.conf** file article describes the available trace options.

Examples

1. To start the **gated** daemon, enter a command similar to the following:

   ```
   startsrc -s gated -a "-tall /var/tmp/gated.log"
   ```

   This command starts the **gated** daemon and logs messages. Messages are sent to the **/var/tmp/gated.log** file.
2. To stop the **gated** daemon normally, enter:

   ```
   stopsrc -s gated
   ```

   This command stops the daemon. The **-s** flag specifies that the subsystem that follows is to be stopped.
3. To get short status from the **gated** daemon, enter:

   ```
   lssrc -s gated
   ```

   This command returns the name of the daemon, the process ID of the daemon, and the state of the daemon (active or inactive).

Files

/etc/gated.pid	Contains the **gated** process ID.
/var/tmp/gated_dump	Specifies the memory dump file.
/var/tmp/gated.log	Specifies the log file for error messages.

Related Information

The **kill** command, **gdc** command, **ospf_monitor** command, and **ripquery** command,

The **routed** daemon.

The **gated.conf** file format.

How to Configure the gated daemon in *Networks and communication management*.

TCP/IP routing, TCP/IP protocols, TCP/IP daemons in *Networks and communication management*.

gdc Command

Purpose

Provides an operational user interface for **gated**.

Syntax

gdc [**-q**] [**-n**] [**-c** *coresize*] [**-f** *filesize*] [**-m** *datasize*] [**-s** *stacksize*] [**-t** *seconds*] *Subcommands*

Description

The **gdc** command provides a user-oriented interface for the operation of the **gated** routing daemon. It provides support for:

- starting and stopping the daemon
- the delivery of signals to manipulate the daemon when it is operating
- the maintenance and syntax checking of configuration files
- for the production and removal of state dumps and core dumps.

The **gdc** command can reliably determine **gated**'s running state and produces a reliable exit status when errors occur, making it advantageous for use in shell scripts which manipulate **gated**. Commands executed using **gdc** and, optionally, error messages produced by the execution of those commands, are logged via the same **syslogd** facility which **gated** itself uses, providing an audit trail of operations performed on the daemon.

Flags

-n	Runs without changing the kernel forwarding table. This is useful for testing, and when operating as a route server which does no forwarding.
-q	Runs quietly. With this flag informational messages which are normally printed to the standard output are suppressed and error messages are logged with **syslogd** instead of being printed to the standard error output. This is convenient when running **gdc** from a shell script.
-t *seconds*	Specifies the time in seconds that **gdc** waits for **gated** to complete certain operations, in particular at termination and startup. By default this value is set to 10 seconds.
-c *coresize*	Sets the maximum size of a core dump a **gated** started with **gdc** produces. This is useful on systems where the default maximum core dump size is too small for **gated** to produce a full core dump on errors.
-f *filesize*	Sets the maximum file size a **gated** started with **gdc** will produce. Useful on systems where the default maximum file dump size is too small for **gated** to produce a full state dump when requested.
-m *datasize*	Sets the maximum size of the data segment of a **gated** started with **gdc**. Useful on systems where the default data segment size is too small for **gated** to run.
-s *stacksize*	Sets the maximum size of stack of a **gated** started with **gdc**. Useful on systems where the default maximum stack size is too small for **gated** to run.

Subcommands

The following subcommands cause signals to be delivered to **gated** for various purpose:

COREDUMP	Sends an abort signal to **gated**, causing it to terminate with a core dump.
dump	Signals **gated** to dump its current state into the file **/var/tmp/gated_dump**.
interface	Signals **gated** to recheck the interface configuration. **gated** normally does this periodically in any event, but the facility can be used to force the daemon to check interface status immediately when changes are known to have occured.
KILL	Causes **gated** to terminate ungracefully.

reconfig Signals **gated** to reread its configuration file, reconfiguring its current state as appropriate.

term Signals **gated** to terminate after shutting down all operating routing protocols gracefully. Executing this command a second time causes **gated** to terminate even if some protocols have not yet fully shut down.

toggletrace Causes tracing to be suspended, and if **gated** is currently tracing to a file, closes the trace file. If **gated** tracing is current suspended, this subcommand causes the trace file to be reopened and tracing initiated. This is useful for moving trace files.

The following subcommands perform operations related to configuration files:

checkconf Check **/etc/gated.conf** for syntax errors. This is usefully done after changes to the configuration file but before sending a **reconfig** signal to the currently running **gated**, to ensure that there are no errors in the configuration which would cause the running **gated** to terminate on reconfiguration. When this command is used, **gdc** issues an informational message indicating whether there were parse errors or not, and if so saves the error output in a file for inspection.

checknew Like **checkconf** except that the **new** configuration file, **/etc/gated.conf+**, is checked instead.

newconf Move the **/etc/gated.conf+** file into place as **/etc/gated.conf**, retaining the older versions of the file as described above. **gdc** will decline to do anything when given this command if the **new** configuration file doesn't exist or otherwise looks suspect.

backout Rotate the configuration files in the **newer** direction, in effect moving the **old** configuration file to **/etc/gated.conf**. The command will decline to perform the operation if **/etc/gated.conf-** doesn't exist or is zero length, or if the operation would delete an existing, non-zero length **/etc/gated.conf+** file.

BACKOUT Perform a **backout** operation even if **/etc/gated.conf+** exists and is of non-zero length.

modeconf Set all configuration files to mode 664, owner root, group system.

createconf If **/etc/gated.conf+** does not exist, create a zero length file with the file mode set to 664, owner root, group system.

The following subcommands provide support for starting and stopping **gated**, and for determining its running state:

running Determine if **gated** is currently running. This is done by checking to see if **gated** has a lock on the file containing its pid, if the pid in the file is sensible and if there is a running process with that pid. Exits with zero status if **gated** is running, non-zero otherwise.

start Start **gated**. The command returns an error if **gated** is already running. Otherwise it executes the **gated** binary and waits for up to the delay interval (10 seconds by default, as set with the **-t** option otherwise) until the newly started process obtains a lock on the pid file. A non-zero exit status is returned if an error is detected while executing the binary, or if a lock is not obtained on the pid file within the specified wait time.

stop Stop **gated**, gracefully if possible, ungracefully if not. The command returns an error (with non-zero exit status) if **gated** is not currently running. Otherwise it sends a terminate signal to **gated** and waits for up to the delay interval (10 seconds by default, as specified with the **-t** option otherwise) for the process to exit. Should **gated** fail to exit within the delay interval it is then signaled again with a second terminate signal. Should it fail to exit by the end of the second delay interval it is signalled for a third time with a kill signal. This should force immediate termination unless something is very broken. The command terminates with zero exit status when it detects that **gated** has terminated, non-zero otherwise.

restart If **gated** is running it is terminated via the same procedure as is used for the **stop** command above. When the previous **gated** terminates, or if it was not running prior to command execution, a new **gated** process is executed using the procedures described for the **start** command above. A non-zero exit status is returned if any step in this procedure appears to have failed.

The following subcommands allow the removal of files created by the execution of some of the commands above:

rmcore	Removes any existing **gated** core dump file.
rmdump	Removes any existing **gated** state dump file.
rmparse	Removes the parse error file generated when a **checkconf** or **checknew** command is executed and syntax errors are encountered in the configuration file being checked.

The following subcommand allows the version information for **gated** to be displayed:

version	Show the version information for **gated**. **gated** cannot already be running at the time this command is executed. No options of **gdc** are used with this command.

By default **gated** obtains its configuration from a file normally named **/etc/gated.conf**. The **gdc** program also maintains several other versions of the configuration file, in particular named:

/etc/gated.conf+	The new configuration file. When **gdc** is requested to install a new configuration file, this file is renamed **/etc/gated.conf**.
/etc/gated.conf-	The old configuration file. When **gdc** is requested to install a new configuration file, the previous **/etc/gated.conf** is renamed to this name.
/etc/gated.conf—	The really old configuration file. **gdc** retains the previous **old** configuration file under this name.

Files

/usr/sbin/gated	The **gated** binary.
/etc/gated.conf	Current **gated** configuration file.
/etc/gated.conf+	Newer configuration file.
/etc/gated.conf-	Older configuration file
/etc/gated.conf—	Much older configuration file
/etc/gated.pid	Where **gated** stores its pid.
/var/tmp/gated_dump	**gated**'s state dump file
/var/tmp/gated.log	Where config file parse errors go.

Related Information

The **gated** Daemon, and **syslogd** Daemon.

---

gencat Command

Purpose

Creates and modifies a message catalog.

Syntax

gencat *CatalogFile SourceFile ...*

Description

The **gencat** command creates a message catalog file (usually ***.cat**) from message text source files (usually ***.msg**). The **gencat** command merges the message text source files, specified by the *SourceFile* parameter, into a formatted message catalog, specified by the *CatalogFile* parameter. After entering messages into a source file, use the **gencat** command to process the source file to create a message catalog. The **gencat** command creates a catalog file if one does not already exist. If the catalog file does exist, the **gencat** command includes the new messages in the catalog file.

You can specify any number of message text source files. The **gencat** command processes multiple source files, one after another, in the sequence specified. Each successive source file modifies the catalog. If the set and message numbers collide, the new message text defined in the *SourceFile* parameter replaces the old message text currently contained in the *CatalogFile* parameter. Message numbers must be in the range of 1 through **NL_MSGMAX**. The set number must be in the range of 1 through **NL_SETMAX**.

The **gencat** command does not accept symbolic message identifiers. You must run the **mkcatdefs** command if you want to use symbolic message identifiers.

> **Note:** Standard output is used if the **-** (dash) character is specified as the *CatalogFile* parameter. Standard input is used if the **-** (dash) character is specified as the *SourceFile* parameter.

Exit Status

This command returns the following exit values:

0 Successful completion.
>0 An error occurred.

Examples

To generate a `test.cat` catalog from the source file `test.msg`, enter:

```
gencat test.cat test.msg
```

The `test.msg` file does not contain symbolic identifiers.

Files

/usr/bin/gencat Contains the **gencat** command.

Related Information

The **dspcat** command, **dspmsg** command, **mkcatdefs** command, **runcat** command.

The **catclose** subroutine, **catgets** subroutine, **catopen** subroutine.

For more information about the Message Facility, see Message Facility Overview in *AIX Version 6.1 National Language Support Guide and Reference*.

---

gencopy Command

Purpose

Allows software products of various packaging formats (installp, RPM, ISMP) to be copied.

Syntax

To Copy Software from Media to Target Location

gencopy -d *Media* [**-t** *TargetLocation*] [**-D**] [**-b** *bffcreateFlags*] [**-U**] [**-X**] **-f** *File* | *CopyList...* | all

To List Software Products and Packages on Media

gencopy -L -d *Media* [**-D**]

Description

The **gencopy** command is the wrapper to the **bffcreate** command. It determines what images must be copied and calls the appropriate command. For RPM, ISMP, or other types of images where the list of required files is unknown, all the files in the subdirectory are copied to the target location.

Flags

-b *bffcreateFlags*	Specifies the following flags that are valid: **l**, **q**, **v**, **w**, and **S**.
-d *Media*	Specifies the device or directory where the install images exist. Media can be a device (**/dev/cd0**, **/dev/rmt0**) or directory.
-D	Specifies debug mode. This flag is for debugging this script. It produces a large quantity of output and should not be used for normal operations.
-f *File*	Specifies a file that contains a list of images to copy to the target location. The **installp**, RPM, and ISMP images should be prefixed with **I:**, **R:**, and **J:**, respectively. Prefix the interim fix packages with an **E:**.
-L	Lists the install packages on the media. This listing is colon separated and contains the following information: file_name:package_name:fileset:V.R.M.F:type:platform:Description bos.sysmgt:bos.sysmgt:bos.sysmgt.nim.client:4.3.4.0:I:R:Network Install Manager - Client Tools bos.sysmgt:bos.sysmgt:bos.sysmgt.smit:4.3.4.0:I:R:System Management Interface Tool (SMIT)
-t *TargetLocation*	Specifies the directory where the installation image files are stored. If the **-t** flag is not specified, the files are saved in the **/usr/sys/inst.images** directory.
-U	Upgrades the directory structure of the destination repository to the current standard, if necessary. The current standard requires images to be organized into subdirectories according to package type and architecture. For example, installp images reside in the *SaveDir*/**installp/ppc** directory. When copying from a source containing this structure, the destination is required to conform. Specifying the **-U** flag permits the **gencopy** command to create the appropriate subdirectory structure in your repository and move any existing images into the appropriate locations. Unless invalid manual copying is performed thereafter, this flag should only need to be used once.
-X	Extends the file system automatically if space is needed.

Security

Attention RBAC users and Trusted AIX users: This command can perform privileged operations. Only privileged users can run privileged operations. For more information about authorizations and privileges, see Privileged Command Database in *Security*. For a list of privileges and the authorizations associated with this command, see the **/etc/security/privcmds** database file.

Example

To copy all of the image from a CD (**/dev/cd0**) to an **LPP_SOURCE** (**/export/lpp_source/500**) use, type:

```
gencopy -d  /dev/cd0 -t  /export/lpp_source/500 all
```

Files

/usr/sbin/gencopy
/usr/sys/inst.data/sys_bundles
/usr/sys/inst.data/user_bundles

Related Information

The **bffcreate** command.

gencore Command

Purpose
Generates a core file for a running process.

Syntax
gencore *ProcessID FileName*

Description
The **gencore** command creates a core file of the process specified by the process ID *ProcessID* without terminating the process. The created core file contains the memory image of the process, which can be used with the **dbx** command for debugging purposes. The core file generated will be named as specified by *FileName* parameter.

The **gencore** command does not create the core file in the location set by the **chcore** or **syscorepath** commands. The core file is placed in the path specified by the *FileName* parameter. If *FileName* specifies only the name of the file, the core file is placed in the current working directory.

Parameters

FileName	Specifies the file name of the core file the **gencore** command creates.
ProcessID	Specifies the process ID of the process from which **gencore** will create a core file.

Exit Status

0 The core file was created successfully.

>0 An Error occurred. A partial core file may be created.

Examples
1. To generate a core file named ″core.1095″ for the process with process ID 1095, enter:
   ```
   gencore 1095 core.1095
   ```
   The creates the core file without terminating the process.

Files

/usr/bin/gencore	Contains the **gencore** command.

Related Information
The **dbx** command, **kill** command.

genfilt Command

Purpose
Adds a filter rule.

Syntax

genfilt -v 4|6 [**-n** *fid*] [**-a D|P|I|L|E|H|S**] **-s** *s_addr* **-m** *s_mask* **[-d** *d_addr*] [**-M** *d_mask*] [**-g Y|N**] [**-c** *protocol*] [**-o** *s_opr*] [**-p** *s_port*] [**-O** *d_opr*] [**-P** *d_port*] [**-r R|L|B**] [**-w I|O|B**] [**-I Y|N**] [**-f Y|N|O|H**] [**-t** *tid*] [**-i** *interface*] [**-D** *description*] [**-e** *expiration_time*] [**-x** *quoted_pattern*] [**-X** *pattern_filename*] [**-C** *antivirus_filename*]

Description

Use the **genfilt** command to add a filter rule to the filter rule table. The filter rules generated by this command are called manual filter rules. IPsec filter rules can be configured using the **genfilt** command, IPsec smit (IP version 4 or IP version 6), or Web-based System Manager in the Virtual Private Network submenu.

Flags

-a *Action*
The following *Action* values are allowed:

- **D** (Deny) blocks traffic.
- **P** (Permit) allows traffic.
- **I** makes this an IF filter rule.
- **L** makes this an ELSE filter rule.
- **E** makes this an ENDIF filter rule.
- **H** makes this a SHUN_HOST filter rule.
- **S** makes this a SHUN_PORT filter rule.

All IF rules most be close with an associated ENDIF rule. These conditional rules can be nested, but correct nesting and scope must be adhered to or the rules will not load correctly with the **mkfilt** command.

-C *antivirus_filename*
Specifies the antivirus file name. The **-C** flag understands some versions of ClamAV Virus Database (http://www.clamav.net).

-c *protocol*
The valid values are: **udp**, **icmp**, **icmpv6**, **tcp**, **tcp/ack**, **ospf**, **ipip**, **esp**, **ah**, and **all**. Value **all** indicates that the filter rule will apply to all the protocols. The protocol can also be specified numerically (between 1 and 252). The default value is **all**.

-D *description*
A short description text for the filter rule. This is an optional flag for static filter rules, it's not applicable to dynamic filter rules.

-d *d_addr*
Specifies the destination address. It can be an IP address or a host name. If a host name is specified, the first IP address returned by the name server for that host will be used. This value along with the destination subnet mask will be compared against the destination address of the IP packets.

-e *expiration_time*
Specifies the expiration time. The expiration time is the amount of time the rule should remain active in seconds. The *expiration_time* does not remove the filter rule from the database. The *expiration_time* relates to the amount of time the filter rule is active while processing network traffic. If no *expiration_time* is specified, then the live time of the filter rule is infinite. If the *expiration_time* is specified in conjunction with a SHUN_PORT (**-a S**) or SHUN_HOST (**-a H**) filter rule, then this is the amount of time the remote port or remote host is denied or shunned once the filter rule parameters are met. If this *expiration_time* is specified independent of a shun rule, then this is the amount of time the filter rule will remain active once the filter rules are loaded into the kernel and start processing network traffic.

-f
Specifies the fragmentation control. This flag specifies that this rule will apply to either all packets (**Y**), fragment headers and unfragmented packets only (**H**), fragments and fragment headers only (**O**), or unfragmented packets only (**N**). The default value is **Y**.

-g
Apply to source routing? Must be specified as **Y** (yes) or **N** (No). If **Y** is specified, this filter rule can apply to IP packets that use source routing. The default value is yes (**Y**). This field only applies to permit rules.

-i *interface*
Specifies the name of IP interface(s) to which the filter rule applies. The examples of the name are: **all**, **tr0**, **en0**, **lo0**, and **pp0**. The default value is **all**.

-l	Specifies the log control. Must be specified as **Y**(yes) or **N** (No). If specified as **Y**, packets that match this filter rule will be included in the filter log. The default value is **N** (no).
-M	Specifies the destination subnet mask. This is used in the comparison of the IP packet's destination address with the destination address of the filter rule.
-m	Specifies the source subnet mask. This is used in the comparison of the IP packet's source address with the source address of the filter rule.
-n	Specifies the filter rule ID. The new rule will be added BEFORE the filter rule you specify. For IP version 4, the ID must be greater than 1 because the first filter rule is a system generated rule and cannot be moved. If this flag is not used, the new rule will be added to the end of the filter rule table.
-O	Specifies the destination port or ICMP code operation. This is the operation that will be used in the comparison between the destination port/ICMP code of the packet with the destination port or ICMP code (**-P** flag). The valid values are: **lt**, **le**, **gt**, **ge**, **eq**, **neq**, and **any**. The default value is **any**. This value must be **any** when the **-c** flag is **ospf**.
-o	Specifies the source port or ICMP type operation. This is the operation that will be used in the comparison between the source port/ICMP type of the packet with the source port or ICMP type(**-p** flag) specified in this filter rule. The valid values are: **lt**, **le**, **gt**, **ge**, **eq**, **neq**, and **any**. The default value is **any**. This value must be **any** when the **-c** flag is **ospf**.
-p	Specifies the source port or ICMP type. This is the value/type that will be compared to the source port (or ICMP type) of the IP packet.
-P	Specifies the destination port/ICMP code. This is the value/code that will be compared to the destination port (or ICMP code) of the IP packet.
-r	Routing. This specifies whether the rule will apply to forwarded packets (**R**), packets destined or originated from the local host (**L**), or both (**B**). The default value is **B**.
-s *s_addr*	Specifies the source address. It can be an IP address or a host name. If a host name is specified, the first IP address returned by the name server for that host will be used. This value along with the source subnet mask will be compared against the source address of the IP packets.
-t	Specifies the ID of the tunnel related to this filter rule. All the packets that match this filter rule must go through the specified tunnel. If this flag is not specified, this rule will only apply to non-tunnel traffic.
-v	Specifies the IP version of the filter rule. Valid values are **4** and **6**.
-w *Direction*	Specifies whether the rule applys to incoming packets (I), outgoing packets (O), or both (B). The default value is B. It is not valid to use the (O) outgoing direction with the **-x**, **-X**, or **-C** pattern options. It is valid to specify the (B) both directions with the pattern options, but only the incoming packets are checked against the packets.
-X *pattern_filename*	Specifies the pattern file name. If more than one patterns are associated with this filter rule, then a pattern file name must be used. The pattern file name must be in the format of one pattern per line. A pattern is an unquoted character string. This file is read once when the filter rules are activated. For more information, see the **mkfilt** command.
-x *pattern*	Specifies the quoted character string or pattern. This string specified is interpreted as an ASCII string unless it is preceded by a `0x`, in which case it is interpreted as a hexadecimal string. The **-x** *pattern* is compared against network traffic.

Security

Attention RBAC users and Trusted AIX users: This command can perform privileged operations. Only privileged users can run privileged operations. For more information about authorizations and privileges, see Privileged Command Database in *Security*. For a list of privileges and the authorizations associated with this command, see the **/etc/security/privcmds** database file.

geninstall Command

Purpose

A generic installer that installs software products of various packaging formats. For example, installp, RPM, SI, and ISMP.

Syntax

geninstall -d *Media* [**-I** *installpFlags*] [**-E** | **-T**] [**-t** *ResponseFileLocation*] [**-e** *LogFile*] [**-p**] [**-F**] [**-Y**] [**-Z**] [**-D**] { **-f** *File* | *Install_List*] | all}

OR

geninstall -u [**-e** *LogFile*] [**-E** | **-T**] [**-t** *ResponseFileLocation*] [**-D**] {**-f** *File* | *Uninstall_List*...}

OR

geninstall -L -d *Media* [**-e** *LogFile*] [**-D**]

Description

Accepts all current **installp** flags and passes them on to **installp**. Some flags (for example, **-L**) are overloaded to mean list all products on the media. Flags that don't make sense for ISMP packaged products are ignored. This allows programs (like NIM) to continue to always send in **installp** flags to **geninstall**, but only the flags that make sense are used.

The **geninstall** command provides an easy way to see what modifications have been made to the configuration files listed in **/etc/check_config.files**. When these files have been changed during a **geninstall** installation or update operation, the differences between the old and new files will be recorded in the **/var/adm/ras/config.diff**. If **/etc/check_config.files** requests that the old file be saved, the old file can be found in the **/var/adm/config** directory.

The **/etc/check_config.files** file can be edited and can be used to specify whether old configuration files that have been changed should be saved (indicated by `s`) or deleted (indicated by `d`), and has the following format:

```
d /etc/inittab
```

A summary of the **geninstall** command's install activity is kept at **/var/adm/sw/geninstall.summary**. This file contains colon-separated lists of filesets installed by **installp** and components installed by ISMP. This is used mainly to provide summary information for silent installs.

Note: Refer to the **README.ISMP** file in the **/usr/lpp/bos** directory to learn more about ISMP-packaged installations and using response files. Also, the **geninstall** command is capable of installing interim fix files containing concurrent updates. Any interim fixes containing concurrent updates must be placed inside a subdirectory called **cupdates** in the directory containing installation images, and the **geninstall** command will install them appropriately.

Flags

-d *Device or Directory*	Specifies the device or directory containing the images to install.
-D	Specifies debug mode. This flag is for debugging this script. It produces a large quantity of output and should not be used for normal operations.

-e *LogFile*	Enables event logging. The **-e** flag enables the user to append certain parts of the **geninstall** command output to the file specified by the *LogFile* variable. The *LogFile* variable must specify an existing, writable file, and the file system in which the file resides must have enough space to store the log. The log file does not wrap.
-E	Creates an ISMP response file recording in the default location, which is the directory containing the product installation files. This option requires running the ISMP installation or uninstallation interactively and completely. The resulting response file will be used to provide the same options on future installations or uninstallations of the same product. Creation of the response file recording will also result in installation or uninstallation of the product.
-f *File*	Specifies a file that contains a list of images to copy to the target location. The **installp**, RPM, and ISMP images should be prefixed with **I:**, **R:**, and **J:**, respectively. Prefix the interim fix packages with an **E:**.
-F	Allows the user to reinstall a package that is already installed, or to install a package that is older than the currently installed version.
-I *installpFlags*	Specifies the **installp** flags to use when calling the **installp** command. The flags that are used during an install operation for **installp** are the **a**, **b**, **c**, **D**, **e**, **E**, **F**, **g**, **I**, **J**, **M**, **N**, **O**, **p**, **Q**, **q**, **S**, **t**, **v**, **V**, **w**, and **X** flags. The **installp** flags that are not used during install are the **C**, **i**, **r** , **z**, **A**, and **l** flags. The **installp** command should be called directly to perform these functions. The **-u**, **-d**, **-L**, and **-f** flags should be given outside the **-I** flag.
-L	Lists the contents of the media. The output format is the same as the **installp -Lc** format, with additional fields at the end for ISMP and RPM formatted products.
-p	Performs a preview of an action by running all preinstallation checks for the specified action.
-t *ResponseFileLocation*	Allows specifying an alternate location for response files or response file templates. The default location is the directory containing the product installation files. This flag can be used to create a response file recording or template in a different location. The *ResponseFileLocation* can either be a file or directory name. If the *ResponseFileLocation* is a directory, it must already exist. If the *ResponseFileLocation* is not an existing directory, it will be assumed that a file name is specified.
-T	Creates an ISMP response file template in the default location, which is the directory containing the product installation files. The resulting template can be used to create a reponse file for future installations or uninstallations of the same product with the desired options. Creation of the response file template will not result in installation or uninstallation of the product.
-u	Performs an uninstall of the specified software. For ISMP products, the uninstaller listed in the vendor database is called, prefixed by a ″J:″.
-Y	Agrees to required software license agreements for software to be installed. This flag is also accepted as an **installp** flag with the **-I** option.
-Z	Tells **geninstall** to invoke the installation in silent mode.

Example

To install all the products on a CD media that is in drive cd0, type:

```
geninstall -d /dev/cd0 all
```

If ISMP images are present on the media, a graphical interface is presented. Any **installp**, SI, or RPM images are installed without prompting, unless the **installp** images are spread out over multiple CDs.

Files

/usr/sbin/geninstall
/usr/sys/inst.data/sys_bundles

/usr/sys/inst.data/user_bundles

Related Information

The **installp** command, **install_wizard** command.

---

genkex Command

Purpose

The **genkex** command extracts the list of kernel extensions currently loaded onto the system and displays the address, size, and path name for each kernel extension in the list.

Syntax

genkex [**-dh**]

Description

For kernel extensions loaded onto the system, the kernel maintains a linked list consisting of data structures called loader entries. A loader entry contains the name of the extension, its starting address, and its size. This information is gathered and reported by the **genkex** command.

Flags

-d	Shows the address and size of the Data section, in addition to the address and size of the Text section.
-h	Displays usage statement.

Examples

To generate the list of loaded kernel extensions, enter:

```
genkex
```

Related Information

The **genkld** command, **genld** command.

Monitoring and tuning commands and subroutines in *Performance management*.

---

genkld Command

Purpose

The **genkld** command extracts the list of shared objects currently loaded onto the system and displays the address, size, and path name for each object on the list.

Syntax

genkld [**-dh**]

Description

For shared objects loaded onto the system, the kernel maintains a linked list consisting of data structures called loader entries. A loader entry contains the name of the object, its starting address, and its size. This information is gathered and reported by the **genkld** command.

Flags

-d	Shows the address and size of the Data section, in addition to the address and size of the Text section.
-h	Displays usage statement.

Examples

To obtain a list of loaded shared objects, enter:

```
genkld
```

Related Information

The **genkex** command, **genld** command.

Monitoring and tuning commands and subroutines in *Performance management*.

---

genld Command

Purpose

The **genld** command collects the list of all processes currently running on the system, and optionally reports the list of loaded objects corresponding to each process.

Syntax

genld [**-h** | **-l** [**-d**]] [**-a** *Area*]

Description

For each process currently running, the **genld** command prints a report consisting of the process ID and name, optionally followed by the list of objects loaded for that process. The object's address and path name are displayed. Members of libraries are shown between brackets. For example, `/usr/lib/libc.a[shr.o]` means `shr.o` is a loaded member of the **libc.a** library.

Note: Unprivileged users can see loaded objects only for their processes.

Flags

-a *Area*	Lists only processes using the shared library area specified by the *Area* parameter.
-d	Shows the address and size of the Data section, in addition to the address and size of the Text section. This option has no effect without the **-l** flag.
-h	Displays the usage statement.
-l	Reports the lists of loaded objects for each process running on the system.

Security

Attention RBAC users and Trusted AIX users: This command can perform privileged operations. Only privileged users can run privileged operations. For more information about authorizations and privileges, see Privileged Command Database in *Security*. For a list of privileges and the authorizations associated with this command, see the **/etc/security/privcmds** database file.

Examples

To obtain the list of loaded objects for each running process, enter:

```
genld -l
```

Related Information

The **genkex** command, **genkld** command.

Monitoring and tuning commands and subroutines in *Performance management*.

---

gennames Command

Purpose

Gathers all the information necessary to run the **filemon** and **netpmon** commands in off-line mode.

Syntax

gennames[**-f**]

Description

The **gennames** command gathers name to address mapping information necessary for the **filemon** and **netpmon** commands to work in off-line mode. The information gathered includes:

- the list of all the loaded kernel extension, similar to what the **genkex** command reports,
- the list of all the loaded shared libraries, similar to what the **genkld** command reports
- the list of all the loaded processes, similar to what the **genld** command reports
- for **/unix** and all kernel extensions and libraries, the output of the **stripnm -z** command is collected

Flags

-f	Collects the device information for physical and logical volumes. It also prints out the virtual file system information used by offline **filemon**.

Security

Attention RBAC users and Trusted AIX users: This command can perform privileged operations. Only privileged users can run privileged operations. For more information about authorizations and privileges, see Privileged Command Database in *Security*. For a list of privileges and the authorizations associated with this command, see the **/etc/security/privcmds** database file.

Examples

To collect information needed for the **filemon** command in off-line mode, type:

```
gennames -f > gen.out
```

Related Information

The **filemon** command, **gensyms** command, **genkex** command, **genld** command, **netpmon** command, **stripnm** command.

Monitoring and tuning commands and subroutines in *Performance management*.

---

gensyms Command

Purpose

Gathers all the information necessary to run the **curt**, **splat**, and **tprof** commands in off-line mode.

Syntax

gensyms [**-ofhs**] [**-k** *kernel*] [**-i** *file*] [**-b** *binary*[,*binary*[,...]]] [**-S** *path*]

Description

The **gensyms** command gathers name to address mapping information necessary for the **curt**, **splat**, and **tprof** commands to work in off-line mode. The information gathered includes:

- the list of all the loaded kernel extension
- the list of all the loaded shared libraries
- the list of all the loaded processes
- for **/unix**, all kernel extensions, libraries, and all object files corresponding to processes, the output of the **stripnm** command is collected

Flags

-b *binary*	Specifies an optional list of binaries for which to find symbols.
-f	Suppresses printing of source file names.
-h	Prints help message.
-i *file*	Reads symbols from specified file.
-k *kernel*	Specifies the name of the kernel image (default **/unix**).
-o	Prints offsets instead of addresses
-s	Finds symbols only for files given by the **-k** and **-b** flags.
-S *path*	Specifies the search path list; it is used to find binaries.

Security

Attention RBAC users and Trusted AIX users: This command can perform privileged operations. Only privileged users can run privileged operations. For more information about authorizations and privileges, see Privileged Command Database in *Security*. For a list of privileges and the authorizations associated with this command, see the **/etc/security/privcmds** database file.

Examples

To collect information needed for the tprof command in off-line mode with the profiling of user program test, type:

```
gensyms > test.syms
```

Related Information

The **curt** command, **gennames** command, **splat** command, **stripnm** command, and **tprof** command.

Monitoring and tuning commands and subroutines in *Performance management*.

---

gentun Command

Purpose

Creates a tunnel definition in the tunnel database.

Syntax

gentun -s *src_host_IP_address* **-d** *dst_host_IP_address* **-v 4**|**6** [**-t** *tun_type*] [**-m** *pkt_mode*] [**-t** *IBM*] [**-t** *manual*] [**-m** *tunnel*] [**-m** *transport*] [**-f** *fw_address*] [**-x** dst_mask]] [**-e** [*src_esp_algo*]] [**-a** [*src_ah_algo*]] [**-p** *src_policy*] [**-A** [*dst_ah_algo*]] [**-P** *dst_policy*] [**-k** *src_esp_key*] [**-h** *src_ah_key*] [**-K** *dst_esp_key*] [**-H** *dst_ah_key*] [**-n** *src_esp_spi*] [**-u** *src_ah_spi*] [**-N** *dst_esp_spi*] [**-U** *dst_ah_spi*] [**-b** *src_enc_mac_algo*] [**-c** *src_enc_mac_key*] [**-B** *dst_enc_mac_algo*] [**-C** *dst_enc_mac_key*] [**-g**] [**-z**] [**-E**]

Description

The **gentun** command creates a definition of a tunnel between a local host and a tunnel partner host. The associated auto-generated filter rules for the tunnel can be optionally generated by this command.

Flags

-a Authentication algorithm, used by source for IP packet authentication. The valid values for **-a** depend on which authentication algorithms have been installed on the host. The list of all the authentication algorithms can be displayed by issuing the **ipsecstat -A** command. The default value is HMAC_MD5 for **manual** tunnels.

-A (**manual** tunnel only) Authentication algorithm, used by destination for IP packet authentication. The valid values for **-A** depend on which authentication algorithms have been installed on the host. The list of all the authentication algorithms can be displayed by issuing the **ipsecstat -A** command. If this flag is not used, the value used by the **-a** flag is used.

-b (**manual** tunnel only) Source ESP Authentication Algorithm (New header format only). The valid values for **-b** depend on which authentication algorithms have been installed on the host. The list of all the authentication algorithms can be displayed by issuing the **ipsecstat -A** command.

-B (**manual** tunnel only) Destination ESP Authentication Algorithm (New header format only). The valid values for **-B** depend on which authentication algorithms have been installed on the host. The list of all the authentication algorithms can be displayed by issuing the **ipsecstat -A** command. If this flag is not used, it is set to the same value as the **-b** flag.

-c (**manual** tunnel only) Source ESP Authentication Key (New header format only). It must be a hexdecimal string started with ″0x″. If this flag is not used, the system will generate one for you.

-C (**manual** tunnel only) Destination ESP Authentication Key (New header format only). It must be a hexdecimal string started with ″0x″. If this flag is not used, it is set to the same value as the **-c** flag.

-d Destination Host IP address. In host-host case, this is the IP address of the destination host interface to be used by the tunnel. In host-firewall-host case, this is the IP address of the destination host behind the firewall. A host name is also valid and the first IP address returned by name server for the host name will be used.

-e Encryption algorithm, used by source for IP packet encryption. The valid values for **-e** depend on which encryption algorithms have been installed on the host. The list of all the encryption algorithms can be displayed by issuing the **ipsecstat -E** command.

-E (**manual** tunnel only) Encryption algorithm, used by destination for IP packet encryption. The valid values for **-E** depend on which encryption algorithms have been installed on the host. The list of all the encryption algorithms can be displayed by issuing the **ipsecstat -E** command. If this flag is not used, the value used by the **-e** flag is used.

-f IP address of the firewall that is between the source and destination hosts. A tunnel will be established between this host and the firewall. Therefore the corresponding tunnel definition must be made on the firewall host. A host name may also be used for this flag and the first IP address returned by the name server for that host name will be used.

-g System auto-generated filter rule flag. If this flag is not used, the command will generate two filter rules for the tunnel automatically. The auto-generated filter rules will allow IP traffic between the two end points of the tunnel to go through the tunnel. If the **-g** flag is specified, the command will only create the tunnel definition, and the user will have to add user defined filter rules to let the tunnel work.

-h This is the AH Key String for a **manual** tunnel. The input must be a hexdecimal string started with ″0x″. If this flag is not used, the system will generate a key using a random number generator.

-H (**manual** tunnel only) The Key String for destination AH. The input must be a hexdecimal string started with ″0x″. If this flag is not used, the system will generate a key using a random number generator.

-k This is the ESP Key String for a **manual** tunnel. It is used by the source to create the tunnel. The input must be a hexdecimal string started with ″0x″. If this flag is not used, the system will generate a key using a random number generator.

-K (**manual** tunnel only) The Key String for destination ESP. The input must be a hexdecimal string started with ″0x″. If this flag is not used, the system will generate a key using a random number generator.

-l Key Lifetime, specified in minutes.

For **manual** tunnels, this value indicates the time of operability before the tunnel expires.

The valid values for **manual** tunnels are 0 - 44640. Value 0 indicates that the **manual** tunnel will never expire. The default value for **manual** tunnels is 480.

-m Secure Packet Mode. This value must be specified as **tunnel** or **transport**. The default value is **tunnel**. Tunnel mode will encapsulate the entire IP packet, while the transport mode only encapsulates the data portion of the IP packet. When generating a host-firewall-host tunnel (for host behind a firewall), the value of **tunnel** must be used for this flag.

The -m flag is forced to use default value (**tunnel**) if the **-f** flag is specified.

-n (**manual** tunnel only) Security Parameter Index for source ESP. This is a numeric value that, along with the destination IP address, identifies which security association to use for packets using ESP. If this flag is not used, the system will generate an SPI for you.

-N (**manual** tunnel only) Security Parameter Index for the destination ESP. It must be entered for a **manual** tunnel if the policy specified in the **-P** flag includes ESP. This flag does not apply to **IBM** tunnels.

-p Source policy, identifies how the IP packet authentication and/or encryption is to be used by this host. If specified as **ea**, the IP packet gets encrypted before authentication. If specified as **ae**, it gets encrypted after authentication, whereas specifying **e** alone or **a** alone corresponds to the IP packet being encrypted only or authenticated only. The default value for this flag will depend on if the **-e** and **-a** flags are supplied. The default policy will be **ea** if either both or neither the **-e** and **-a** flags are supplied. Otherwise the policy will reflect which of the **-e** and **-a** flags were supplied.

-P (**manual** tunnel only) Destination policy, identifies how the IP packet authentication and/or encryption is to be used by destination. If specified as **ea**, the IP packet gets encrypted before authentication. If specified as **ae**, it gets encrypted after authentication, whereas specifying **e** or **a** corresponds to the IP packet being encrypted only or authenticated only. The default policy will be **ea** if either both or neither the **-E** and **-A** flags are supplied. Otherwise, the policy will reflect which of the **-E** and **-A** flags were specified.

-s Source Host IP address, IP address of the local host interface to be used by the tunnel. A host name is also valid and the first IP address returned by name server for the host name will be used.

-t Type of the tunnel. Must be specified as **manual**.

The initial tunnel key and any subsequent key updates need to be performed manually when using the **manual** tunnel. Once a key is installed manually, that same key is used for all tunnel operations until it is changed manually.

The **manual** tunnel value should be selected when you want to construct a tunnel with a non-IBM IP Security host or any IP version 6 end-point, where the end-point either supports RFCs 1825-1829 or the IETF drafts for the new IP Security encapsulation formats for IP tunnels.

-u (**manual** tunnel only) Security Parameter Index for source AH. Use SPI and the destination IP address to determine which security association to use for AH. If this flag is not used, the value of the -n SPI will be used.

-U (**manual** tunnel only) Security Parameter Index for the destination AH. If this flag is not used, the -N spi will be used.

-v The IP version for which the tunnel is created. For IP version 4 tunnels, use the value of **4**. For IP version 6 tunnels, use the value of **6**.

-x Network mask for the secure network behind a firewall. The Destination host is a member of the secure network. The combination of **-d** and **-x** allows the source host to communicate with multiple hosts in the secure network through the source-firewall tunnel, which must be in tunnel mode.

This flag is valid only when the **-f** flag is used.

-y (**manual** tunnel only) Replay prevention flag. Replay prevention is valid only when the ESP or AH header is using the new header format (see the **-z** flag). The valid values for the **-y** flag are Y (yes) and N (no). All encapsulations that are used in this tunnel (AH, ESP, sending, and receiving) will use the replay field if the value of this flag is Y. The default value is N.

-z (**manual** tunnel only) New header format flag. The new header format preserves a field in the ESP and AH headers for replay prevention and also allows ESP authentication. The replay field will only be used when the replay flag (**-y**) is set to Y. The valid values for the **-z** flag are Y (yes) and N (no). The default value when the **-z** flag is not used depends on the algorithms you've chosen for the tunnel. It will default to N unless either an algorithm other than KEYED_MD5 is used for either the **-a** or **-A** flags, or if the **-b** or **-B** flags are used.

Security

Attention RBAC users and Trusted AIX users: This command can perform privileged operations. Only privileged users can run privileged operations. For more information about authorizations and privileges, see Privileged Command Database in *Security*. For a list of privileges and the authorizations associated with this command, see the **/etc/security/privcmds** database file.

Related Information

The **chtun** command, **exptun** command, **imptun** command, **lstun** command, **mktun** command, and **rmtun** command.

---

genxlt Command

Purpose

Generates a code set conversion table for use by the **Iconv** library.

Syntax

genxlt [OutputFile]

Description

The **genxlt** command reads a source code set conversion table file from standard input and writes the compiled version to the file specified by the *OutputFile* parameter. If a value is not specified for the *OutputFile* parameter, standard output is used. The source code set conversion table file contains directives that are acted upon by the **genxlt** command to produce the compiled version.

The format of a code set conversion table source file is:

- Lines whose initial nonwhite space character is the # (pound sign) are treated as comment lines.
- Null lines and lines consisting only of white-space characters are treated as comment lines.
- Non-comment lines have to be of the following form:

```
%token <blank>   # <tab> and <space>
%token <hex>     # <zero>, <one>, <two>, <three>, <four>,
                 # <five>, <six>, <seven>, <eight>, <nine>,
                 # <a>, <b>, <c>, <d>, <e>, <f>,
                 #  <A>, <B>, <C>, <D>, <E>, <F>,
%token <any>     # any character but '\n'
line    :  offset  blank  value  blank  comment  '\n'
        | 'SUB'  blank  value  blank  comment  '\n'
        ;

blank    : <blank>
        | blank <blank>
        ;

offset   :  '0x'  <hex>
        |  offset  <hex>
        ;
```

```
value    :  offset
         |  'invalid'
         |  'substitution'
         ;


comment  :  '#'  <any>
         |  comment  <any>
         ;
```

A line where the offset is 'SUB' is used to specify the default substitution character.

If the table is set to 'substitution', the **iconv** converter using this table uses the SUB value for this offset.

If the value is set to 'invalid', the **iconv** converter using this table returns error for its offset.

If the offset is found in the source code set conversion table file multiple times, the last entry is used in the compilation of the translation table.

The offset and value must be in the range of 0x00 through 0xff, inclusive.

The following is an excerpt of a code set conversion table:

```
SUB     0x1a     substitute character
0x80    0xc7     C  cedilla
0x81    0xfc     u  diaeresis
0x82    0xe9     e  acute
0x83    0xe2     a  circumflex
0x84    0xe4     a  diaeresis
0x85    0x40     a  grave
0x9F            substitution
0xff            invalid
```

If successful, the **genxlt** command exits with a value of 0. If the output file cannot be opened, the **genxlt** command is unsuccessful and exits with a value of 1. If a syntax error is detected in the input stream, the **genxlt** command will exit immediately with a value of 2, and write to standard error the line numbers where the syntax error occurred.

The name of the file generated by the **genxlt** command must follow the naming convention below in order for the **iconv** subsystem to recognize it as a conversion file:

```
fromcode: "IBM-850"
tocode: "ISO8859-1"
conversion table file: "IBM-850_ISO8859-1"
```

The conversion table file name is formed by concatenating the tocode file name onto the fromcode file name, with an underscore between the two.

Example

To generate a non-English, user-defined code set conversion table, enter:

```
cp  /usr/lib/nls/loc/iconvTable/ISO8859-1_IBM-850_src  $HOME
vi  $HOME/ISO8859-1_IBM-850_src
genxlt  <  $HOME/ISO8859-1_IBM-850_src  >  cs1_cs2
```

Related Information

The **iconv** command.

The **iconv_open** subroutine, **iconv** subroutine, and **iconv_close** subroutine provide a method to use the conversion service from within a program.

National Language Support and Converters Overview for Programming in *AIX Version 6.1 National Language Support Guide and Reference*.

get Command

Purpose

Creates a specified version of a SCCS file.

Syntax

To Get Read-Only Versions of SCCS Files

get [**-g**] [**-m**] [**-n**] [**-p**] [**-s**] [**-c** *Cutoff*] [**-i** *List*] [**-r** *SID*] [**-t**] [**-x** *List*] [**-w** *String*] [**-l** [**p**]] [**-L**] *File* ...

To Get Editable Versions of SCCS Files

get [**-e**] [**-k**] [**-b**] [**-s**] [**-c** *Cutoff*] [**-i** *List*] [**-r** *SID*] [**-t**] [**-x** *List*] [**-l** [**p**]] [**-L**] *File* ...

Description

The **get** command reads a specified version of the Source Code Control System (SCCS) file and creates an ASCII text file according to the specified flags. The **get** command then writes each text file to a file having the same name as the original SCCS file but without the **s.** prefix (the **g-file**).

Flags and files can be specified in any order, and all flags apply to all named files. If you specify a directory for the *File* parameter, the **get** command performs the requested actions on all files in the directory that begin with the **s.** prefix. If you specify a **-** (minus sign) for the *File* parameter, the **get** command reads standard input and interprets each line as the name of an SCCS file. The **get** command continues to read input until it reads an end-of-file character.

If the effective user has write permission in the directory containing the SCCS files but the real user does not, then only one file can be named when the **-e** flag is used.

Note: The **get** command supports the Multibyte Character Set (MBCS) for the file name and string data specified with the **w** flag.

Getting Read-Only File Versions

The **get** command creates both read-only versions and editable versions of a file. Read-only versions of files should be used if the application does not require changes to the file contents. Read-only versions of source code files can be compiled. Text files can be displayed or printed from read-only versions.

The difference between an editable and a read-only version is important when using identification keywords. *Identification keywords* are symbols expanded to some text value when the **get** command retrieves the file as read-only. In editable versions, keywords are not expanded. Identification keywords can appear anywhere in an SCCS file. See the **prs** command for further information on identification keywords.

SCCS Files

In addition to the file with the **s.** prefix (the **s-file**), the **get** command creates several auxiliary files: the **g-file**, **l-file**, **p-file**, and **z-file**. These files are identified by their tag, which is the letter before the hyphen. The **get** program names auxiliary files by replacing the leading **s.** in the SCCS file name with the appropriate tag, except for the **g-file**, which is named by removing the **s.** prefix. So, for a file named **s.sample**, the auxiliary file names would be **sample**, **l.sample**, **p.sample**, and **z.sample**.

These files serve the following purposes:

s-file Contains the original file text and all the changes (deltas) made to the file. It also includes information about who can change the file contents, who has made changes, when those changes were made, and the nature of changes made. You cannot edit this file directly because it is read-only. However, it contains the information needed by the SCCS commands to build the **g-file**, which you can edit.

g-file An ASCII text file that contains the text of the SCCS file version that you specify with the **-r** flag (or the latest trunk version by default). You can edit this file directly. When you have made all your changes and want to make a new delta to the file, you can then run the **delta** command on the file. The **get** command creates the **g-file** in the current directory.

Whenever it runs the **get** command creates a g-file, unless the **-g** flag or the **-p** flag is specified. The real user owns it (not the effective user). If you do not specify the **-k** or **-e** flag, the file is read-only. If the **-k** or **-e** flag is specified, the owner has write permission for the **g-file**. You must have write permission in the current directory to create a **g-file**.

l-file The **get** command creates the **l-file** when the **-l** flag is specified. The **l-file** is a read-only file. It contains a table showing which deltas were applied in generating the **g-file**. You must have write permission in the current directory to create an **l-file**. Lines in the **l-file** have the following format:

- A blank character if the delta was applied; otherwise, an asterisk.
- A blank character if the delta was applied, or was not applied and ignored. An asterisk appears if the delta was not applied and not ignored.
- A code indicating a special reason why the delta was or was not applied:

  Blankspace
  Included or excluded usually

  I Included using the **-i** flag

  X Excluded using the **-x** flag

  C Cut off using the **-c** flag
- The SID.
- The date and time the file was created.
- The login name of person who created the delta.

Comments and Modification Requests (MR) data follow on subsequent lines, indented one horizontal tab character. A blank line ends each entry. For example, for a delta cutoff with the **-c** flag, the entry in the l-file might be:

```
**C 1.3 85/03/13 12:44:16 pat
```

and the entry for the initial delta might be:

```
1.1 85/02/27 15:42:20 pat
date and time created 85/02/27 15:42:20 by pat
```

p-file The **get** command creates the **p-file** when the **-e** or **-k** flag is specified. The **p-file** passes information resulting from a **get -e** command to a **delta** command. The **p-file** also prevents a subsequent execution of a **get -e** command for the same SID until a **delta** command is run or the joint edit key letter (**j**) is set in the SCCS file. The **j** key letter allows several **get** commands to be run on the same SID. The **p-file** is created in the directory containing the SCCS *File*. To create a **p-file** in the SCCS directory, you must have write permission in that directory. The permission code of the **p-file** is read-only to all but its owner, and it is owned by the effective user. The **p-file** should not be directly edited by the owner. The **p-file** contains:

- Current SID
- SID of new delta to be created
- User name
- Date and time of the **get** command
- **-i** flag, if present
- **-x** flag, if present

The **p-file** contains an entry with the preceding information for each pending delta for the file. No two lines have the same new delta SID.

z-file The **z-file** is a lock mechanism against simultaneous updates. The **z-file** contains the binary process number of the **get** command that created it. This file is created in the directory containing the SCCS file and exists only while the **get** command is running.

When you use the **get** command, it displays the SID being accessed and the number of lines created from the SCCS file. If you specify the **-e** flag, the SID of the delta to be made appears after the SID is accessed and before the number of lines created. If you specify more than one file, a directory, or standard input, the **get** command displays the file name before each file is processed. If you specify the **-i** flag, the **get** command lists included deltas below the word `Included`. If you specify the **-x** flag, the **get** command lists excluded deltas below the word `Excluded`.

The following table illustrates how the **get** command determines both the SID of the file it retrieves and the pending SID. The SID Specified column shows various ways the SID can be specified with the **-r** flag. The first column also illustrates various conditions that can exist, including whether or not the **-b** flag is used with the **get -e** command. The SID Retrieved column indicates the SID of the file that makes up the **g-file**. The SID of Delta to Be Created column indicates the SID of the version that will be created when the **delta** command is applied.

SID Determination

SID Specified	SID Retrieved	SID of Delta to Be Created
none [1] **-b Used?** no **Other Conditions** R defaults to mR [2]	mR.mL	mR.(mL+1)
none [1] **-b Used?** yes **Other Conditions** R defaults to mR	mR.mL	mR.mL.(mB+1).1
R **-b Used?** no **Other Conditions** R>mR	mR.mL	R.1 [3]

SID Determination

SID Specified	SID Retrieved	SID of Delta to Be Created
R **-b Used?** no **Other Conditions** R=mR	mR.mL	mR.(mL+1)
R **-b Used?** yes **Other Conditions** R>mR	mR.mL	mR.mL.(mB+1).1
R **-b Used?** yes **Other Conditions** R=mR	mR.mL	mR.mL.(mB+1).1
R **-b Used?** N/A **Other Conditions** R<mR and R does not exist	hR.mL [4]	hR.mL.(mB+1) .1
R **-b Used?** N/A **Other Conditions** Trunk successor in release > R and R exists	R.mL	R.mL.(mB+1).1
R.L. **-b Used?** no **Other Conditions** No trunk successor	R.L.	R.(L+1)
R.L. **-b Used?** yes **Other Conditions** No trunk successor	R.L.	R.L(mB+1).1
R.L. **-b Used?** N/A **Other Conditions** Trunk successor in release > or = R	R.L.	R.L.(mB+1).1

SID Determination

SID Specified	SID Retrieved	SID of Delta to Be Created
R.L.B. **-b Used?** no **Other Conditions** No branch successor	R.L.B.mS	R.L.B.(mS+1)
R.L.B. **-b Used?** yes **Other Conditions** No branch successor	R.L.B.mS	R.L.(mB+1).1
R.L.B.S. **-b Used?** no **Other Conditions** No branch successor	R.L.B.S.	R.L.B.(S+1)
R.L.B.S. **-b Used?** yes **Other Conditions** No branch successor	R.L.B.S.	R.L.(mB+1).1
R.L.B.S. **-b Used?** N/A **Other Conditions** Branch successor	R.L.B.S.	R.L.(mB+1).1
Note: In the SID Determination table, the letters R, L, B, and S are the release, level, branch, and sequence components of the SID. The letter *m* signifies maximum.		

[1] Applies only if the **-d** (default SID) flag is not present in the file (see the **admin** command).

[2] The mR indicates the maximum existing release.

[3] Forces creation of the first delta in a new release.

[4] The hR is the highest existing release lower than the specified, nonexistent release R.

Identification Keywords

Identifying information is inserted into the text retrieved from the SCCS file by replacing identification keywords with their value wherever they occur. The following keywords may be used in the text stored in an SCCS file:

Keyword	Value
%M%	Module name: either the value of the **m** flag in the file, or, if absent, the name of the SCCS file with the **s.** removed.
%I%	SCCS identification (SID) (*%R%.%L%* or *%R%.%L%.%B%.%S%*) of the retrieved text.
%R%	Release.

Keyword	Value
%L%	Level.
%B%	Branch.
%S%	Sequence.
%D%	Current date, formatted as *YY/MM/DD.*
%H%	Current date, formatted as *MM/DD/YY.*
%T%	Current time, formatted as *HH:MM:SS.*
%E%	Date newest applied delta was created, formatted as *YY/MM/DD.*
%G%	Date newest applied delta was created, formatted as *MM/DD/YY.*
%Y%	Module type: value of the **t** flag in the SCCS file.
%F%	SCCS file name.
%P%	SCCS absolute path name.
%Q%	The value of the **-q** flag in the file.
%C%	Current line number. This keyword is intended for identifying messages output by the program, such as `this should not have happened` error messages. The **%C%** is not intended to be used on every line to provide sequence numbers.
%Z%	The four-character string `@(#)` recognizable by *what.*
%W%	A shorthand notation for constructing *what* strings: `%W% = %Z%%M%<tab>%I%`
%A%	Another shorthand notation for constructing *what* strings: `%A% = %Z%%Y% %M% %I%%Z%`

Flags

-b Specifies that the delta to be created should have an SID in a new branch. The new SID is numbered according to the rules given in the SID determination table. You can use the **-b** flag only with the **-e** flag. It is only necessary when you want to branch from a leaf delta (a delta without a successor). Attempting to create a delta at a nonleaf delta automatically results in a branch, even if the **b** header flag is not set. If you do not specify the **b** header flag in the SCCS file, the **get** command ignores the **-b** flag because the file does not allow branching.

-c *Cutoff* Specifies a cutoff date and time, in the form *YY[MM[DD[HH[MM[SS]]]]]*. The **get** command includes no deltas to the SCCS file created after the specified cutoff in the g-file. The values of any unspecified items in the *Cutoff* variable default to their maximum allowable values. Thus, a cutoff date and time specified with only the year (YY) would specify the last month, day, hour, minute, and second of that year. Any number of nonnumeric characters can separate the two-digit items of the *Cutoff* variable date and time. This allows you to specify a date and time in a number of ways, as follows:

```
-c85/9/2,9:00:00
-c"85/9/2 9:00:00"
"-c85/9/2 9:00:00"
```

-e Indicates that the g-file being created is to be edited by the user applying the **get** command. The changes are recorded later with the **delta** command. The **get -e** command creates a p-file that prevents other users from issuing another **get -e** command and editing a second g-file on the same SID before the **delta** command is run. The owner of the file can override this restriction by allowing joint editing on the same SID through the use of the **admin** command with the **-fj** flag. Other users, with permission, can obtain read-only copies by using the **get** command without the **-e** flag. The **get -e** command enforces SCCS file protection specified with the ceiling, floor, and authorized user list in the SCCS file. See the **admin** command.
Note: If you accidentally ruin the g-file created using the **get -e** command, you can recreate the file with the **get -k** command.

-g Suppresses the actual creation of the **g-file**. Use the **-g** flag primarily to create an **l-file** or to verify the existence of a particular SID. Do not use it with the **-e** flag.

-i *List* Specifies a list of deltas to be included in the creation of a **g-file**. The SID list format consists of a combination of individual SIDs separated by commas and SID ranges indicated by two SIDs separated by a hyphen. You can specify the same SIDs with either of the following command lines:

```
get -e -i1.4,1.5,1.6 s.file
get -e -i1.4-1.6 s.file
```

You can specify the SCCS identification of a delta in any form shown in the SID Specified column of the previous table. The **get** command interprets partial SIDs as shown in the SID Retrieved column.

-k Suppresses replacement of identification keywords in the **g-file** by their value. The **-k** flag is implied by the **-e** flag. If you accidentally ruin the **g-file** created using the **get -e** command, you can recreate the file by reissuing the **get** command with the **-k** flag instead of the **-e** flag.

-l[**p**] Writes a delta summary to an l-file. If you specify **-lp**, the delta summary is written to standard output, and the **get** command does not create the **l-file**. Use this flag to determine which deltas were used to create the g-file currently in use. See the **sccsfile** file for the format of the **l-file**. See also the **-L** flag.

-L Writes a delta summary to standard output. Specifying the **-L** flag is the same as using the **-lp** flag.

-m Writes before each line of text in the **g-file** the SID of the delta that inserted the line into the SCCS file. The format is:

```
SID tab line of text
```

-n Writes the value of the **%M%** keyword before each line of text in the **g-file**. The format is the value of **%M%**, followed by a horizontal tab, followed by the text line. When both the **-m** and **-n** flags are used, the format is:

```
%M% value tab SID tab line of text
```

-p Writes the text created from the SCCS file to standard output and does not create a g-file. All informative output usually sent to standard output is sent to standard error, unless you specify the **-s** flag with the **-p** flag. In this case, output usually sent to standard output does not appear anywhere.

-r *SID* Specifies the SCCS identification string (SID) of the SCCS file version to be created. The SID determination table shows the version of the created file and the SID of the pending delta as functions of the specified SID.

-s Suppresses all output usually written to standard output. Error messages (written to standard error output), remain unaffected.

-t Accesses the most recently created delta in a given release or for a given release and level.

-w *String* Substitutes the *String* value for the **%W%** keyword in g-files not intended for editing.

-x *List* Excludes the specified list of deltas in the creation of the **g-file**. See the **-i** flag for the SID list format.

Exit Status

This command returns the following exit values:

0 Successful completion.

>0 An error occurred.

Examples

The following descriptions and examples illustrate the differences between read-only and editable versions of files.

1. To print the current date and SID in a file, put the following symbols in the file:

   ```
   %H% %I%
   ```

   %H% is the symbol for the current date and **%I%** is the symbol for the SID. When the **get** command retrieves a file as editable, it leaves the symbols in the file and does not perform text value substitution.

2. The following example of the **get** command builds the version with the highest SID, because the example does not specify a version of the file:

```
$ ls
s.test.c
$ get s.test.c
3.5
59 lines
$ ls
s.test.c test.c
```

3. In the next two examples, the **-r** flag specifies which version to get:

```
$ get -r1.3 s.test.c
1.3
67 lines

$ get -r1.3.1.4 s.test.c
1.3.1.4
50 lines
```

4. If you specify just the release number of the SID, the **get** command finds the file with the highest level within that release number.

```
$ get -r2 s.test.c
2.7
21 lines
```

5. If the SID specified is greater than the highest existing SID, the **get** command gets the highest existing SID. If the SID specified is lower than the lowest existing SID, SCCS writes an error message. In the following example, release 7 is the highest existing release:

```
$ get -r9 s.test.c
7.6
400 lines
```

6. The **-t** flag gets the top version in a given release or level. The top version is the most recently created delta, independent of its location. In the next example, the highest existing delta in release 3 is 3.5, while the most recently created delta is 3.2.1.5.

```
$ get -t -r3 s.test.c
3.2.1.5
46 lines
```

7. The previous examples use the **get** command to get a read-only file. To create a copy of the file that can be edited and used to create a new delta, use the **get** command with the **-e** flag. Use **unget** to undo the effect of the **get -e** command and discard any changes made to the file before a delta is created. The following example shows how to use the **-e** flag:

```
$ ls
s.test.c
$ get -e s.test.c
1.3
new delta 1.4
67 lines
$ ls
p.test.c s.test.c test.c
```

   The working file is `test.c`. If you edit this file and save the changes with the **delta** command, SCCS creates a new delta with an SID of `1.4`. The file `p.test.c` is a temporary file used by SCCS to keep track of file versions.

   In the previous example, you could have used the **-r** flag to get a specific version. Assuming release `1` is the highest existing release and that delta `1.3` already exists and is the highest delta in release, the following three uses of the **get** command are equivalent:

```
$ get -e s.test.c
$ get -e -r1 s.test.c
$ get -e -r1.3 s.test.c
```

8. To start using a new (higher in value) release number, get the file with the **-r** flag and specify a release number greater than the highest existing release number. In the next example, release 2 does not yet exist:

```
$ get -e -r2 s.test.c
1.3
new delta 2.1
67 lines
```

   Notice that the **get** command indicates the version of the new delta that will be created if the **delta** command stores changes to the SCCS file.

9. To create a branch delta, use the **-r** flag and specify the release and level where the branch occurs. In the next example, deltas `1.3` and `1.4` already exist.

```
$ get -e -r1.3 s.test.c
1.3
new delta 1.3.1.1
67 lines
```

   Creates deltas on branches using the same methods.

   To edit a file, get the file version using the **get -e** command and save the changes with the **delta** command. Several different editable versions of an SCCS file can exist as long as each one is in a different directory. If you try to put duplicates of an editable file version into a directory (using the **get** command) without using the **delta** command, SCCS writes an error message.

   To get the same editable file version more than once, set the **j** header flag in the SCCS file with the **admin** command. Set the **j** option by using the **-f** flag. You can then get the same SID several times from different directories, creating a separate file for each **get** command. Although the files originate from a single SID, SCCS gives each of them a unique new SID.

10. In the following example, the **pwd** command displays the current directory. Then the **j** option is set with the **admin** command:

    Note: You must have write access in both directories to issue the commands in this example.

```
$ pwd
/home/marty/sccs
$ admin -fj s.test.c
```

11. Then use the **get** command to retrieve the latest version of the file:

    Note: You must have write access in both directories to issue the commands in this example.

```
$ get -e s.test.c
1.1
new delta 1.2
5 lines
```

12. Change to the /home/new directory, and issue the **get** command again.

    Note: You must have write access in both directories to issue the commands in this example.

```
$ cd /home/new
$ get -e /home/marty/sccs/s.test.c
1.2
new delta 1.1.1.1
5 lines
```

    Notice that SCCS creates two deltas, `1.2` and `1.1.1.1`, from the single original file version of `1.1`. Look at the **p.test.c** file. It shows a separate entry for each version currently in use. The **p.test.c** file remains in the directory until you take care of both file versions with either the **delta** command or the **unget** command.

Files

/usr/bin/get Contains the **get** command.

Related Information

The **admin** command, **delta** command, **prs** command, and **sact** command, **sccshelp** command, **unget** command, **what** command.

The **sccsfile** file format in *AIX Version 6.1 Files Reference*.

List of SCCS Commands in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

Source Code Control System (SCCS) Overview in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

---

getconf Command

Purpose

Writes system configuration variable values to standard output.

Syntax

getconf [**-v** *specification*] [*SystemwideConfiguration* | *PathConfiguration PathName*] [*DeviceVariable DeviceName*]

getconf -a

Description

The **getconf** command, invoked with the *SystemwideConfiguration* parameter, writes the value of the variable, as specified by the *SystemwideConfiguration* parameter, to standard output.

The **getconf** command, invoked with the *PathConfiguration* and *Pathname* parameters, writes the value of the variable, as specified by the *PathConfiguration* parameter for the path specified by the *PathName* parameter, to standard output.

The **getconf** command, invoked with the **-a** flag, writes the values of all system configuration variables to standard output.

The **getconf** command, invoked with the *DeviceVariable* and *DeviceName* parameters, writes the value of the disk device name or location, for the device path specified by the *DeviceName* parameter, to the standard output.

If the specified variable is defined on the system and its value is described to be available from the **confstr** subroutine, the value of the specified variable is written in the following format:

```
"%s\n", <value>
```

Otherwise, if the specified variable is defined on the system, its value is written in the following format:

```
"%d\n", <value>
```

If the specified variable is valid but undefined on the system, the following is written to standard output:

```
"undefined\n"
```

If the variable name is invalid or an error occurs, a diagnostic message is written to the standard error.

Flags

-v *specification*	Indicates a specific specification and version for which configuration variables are to be determined. If this flag is not specified, the values returned will correspond to an implementation default XBS5 conforming compilation environment.
-a	Writes the values of all system configuration variables to standard output.

Parameters

PathName	Specifies a path name for the *PathConfiguration* parameter.
SystemwideConfiguration	Specifies a system configuration variable.
PathConfiguration	Specifies a system path configuration variable.
DeviceName	Specifies the path name of a device.
DeviceVariable	Specifies a device variable.

When the symbol listed in the first column of the following table is used as the **system_var** operand, **getconf** will yield the same value as **confstr** when called with the value in the second column:

Note: The **_CS_AIX_ARCHITECTURE** and **_CS_AIX_BOOTDEV** variables, used as parameters to **confstr**, are available only to the root user.

system_var	confstr Name Value
BOOT_DEVICE	_CS_AIX_BOOTDEV
MACHINE_ARCHITECTURE	_CS_AIX_ARCHITECTURE
MODEL_CODE	_CS_AIX_MODEL_CODE
PATH	_CS_PATH
XBS5_ILP32_OFF32_CFLAGS	_CS_XBS5_ILP32_OFF32_CFLAGS
XBS5_ILP32_OFF32_LDFLAGS	_CS_XBS5_ILP32_OFF32_LDFLAGS
XBS5_ILP32_OFF32_LIBS	_CS_XBS5_ILP32_OFF32_LIBS
XBS5_ILP32_OFF32_LINTFLAGS	_CS_XBS5_ILP32_OFF32_LINTFLAGS
XBS5_ILP32_OFFBIG_CFLAGS	_CS_XBS5_ILP32_OFFBIG_CFLAGS
XBS5_ILP32_OFFBIG_LDFLAGS	_CS_XBS5_ILP32_OFFBIG_LDFLAGS
XBS5_ILP32_OFFBIG_LIBS	_CS_XBS5_ILP32_OFFBIG_LIBS
XBS5_ILP32_OFFBIG_LINTFLAGS	_CS_XBS5_ILPBIG_OFF32_LINTFLAGS
XBS5_LP64_OFF64_CFLAGS	_CS_XBS5_LP64_OFF64_CFLAGS
XBS5_LP64_OFF64_LDFLAGS	_CS_XBS5_LP64_OFF64_LDFLAGS
XBS5_LP64_OFF64_LIBS	_CS_XBS5_LP64_OFF64_LIBS
XBS5_LP64_OFF64_LINTFLAGS	_CS_XBS5_LP64_OFF64_LINTFLAGS
XBS5_LPBIG_OFFBIG_CFLAGS	_CS_XBS5_LPBIG_OFFBIG_CFLAGS
XBS5_LPBIG_OFFBIG_LDFLAGS	_CS_XBS5_LPBIG_OFFBIG_LDFLAGS
XBS5_LPBIG_OFFBIG_LIBS	_CS_XBS5_LPBIG_OFFBIG_LIBS
XBS5_LPBIG_OFFBIG_LINTFLAGS	_CS_XBS5_LPBIG_OFFBIG_LINTFLAGS

Environment Variables

The following environment variables affect the execution of **getconf**:

LANG	Provide a default value for the internationalisation variables that are unset or null. If LANG is unset or null, the corresponding value from the implementation-dependent | default locale will be used. If any of the internationalisation variables contains an invalid setting, the utility will behave as if none of the variables had been defined.
LC_CALL	If set to a non-empty string value, override the values of all the other internationalisation variables.
LC_CTYPE	Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single- as opposed to multi-byte characters in arguments).
LC_MESSAGES	Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error.
NLSPATH	Determine the location of message catalogues for the processing of LC_MESSAGES.

Systemwide Configuration Variables

The *SystemwideConfiguration* parameter specifies system configuration variables whose values are valid throughout the system. There are two kinds of system configuration variables:

- Systemwide configuration variables
- System standards configuration variables

Systemwide Configuration Variables

Systemwide configuration variables contain the minimum values met throughout all portions of the system. The following list defines the systemwide configuration variables used with the **getconf** command:

_CS_PATH	Value for the **PATH** environment variable used to find commands.
ARG_MAX	Maximum length, in bytes, of the arguments for one of the **exec** subroutines, including environment data.
BC_BASE_MAX	Maximum value allowed for the **obase** variable with the **bc** command.
BC_DIM_MAX	Maximum number of elements permitted in an array by the **bc** command.
BC_SCALE_MAX	Maximum value allowed for the **scale** variable with the **bc** command.
BC_STRING_MAX	Maximum length of a string constant accepted by the **bc** command.
CHARCLASS_NAME_MAX	Maximum number of bytes in a character class name.
CHAR_BIT	Number of bits in a type **character**.
CHAR_MAX	Maximum value of a type **character**.
CHAR_MIN	Minimum value of a type **character**.
CHILD_MAX	Maximum number of simultaneous processes for each real user ID.
CLK_TCK	Number of clock ticks per second returned by the **time** subroutine.
COLL_WEIGHTS_MAX	Maximum number of weights that can be assigned to an entry in the **LC_COLLATE** locale stanza in a locale-definition file.
CS_PATH	Value of the **PATH** environment variable used to find commands.
EXPR_NEST_MAX	Maximum number of expressions that can be nested within parentheses by the **expr** command.
INT_MAX	Maximum value of a type **int**.
INT_MIN	Minimum value of a type **int**.
LINE_MAX	Maximum length, in bytes, of a command's input line (either standard input or another file) when the utility is described as processing text files. The length includes room for the trailing new-line character.
LONG_BIT	Number of bits in a type **long int**.
LONG_MAX	Maximum value of a type **long int**.
LONG_MIN	Minimum value of a type **long int**.
MB_LEN_MAX	Maximum number of bytes in a character for any supported locale.
NGROUPS_MAX	Maximum number of simultaneous supplementary group IDs for each process.
NL_ARGMAX	Maximum value of digit in calls to the **printf** and **scanf** subroutines.
NL_LANGMAX	Maximum number of bytes in a LANG name.

NL_MSGMAX	Maximum message number.
NL_NMAX	Maximum number of bytes in an N-to-1 collation mapping.
NL_SETMAX	Maximum set number.
NL_TEXTMAX	Maximum number of bytes in a message string.
NZERO	Default process priority.
OPEN_MAX	Maximum number of files that one process can have open at one time.
PATH	Sequence of colon-separated path prefixes used to find commands.
RE_DUP_MAX	Maximum number of repeated occurrences of a regular expression permitted when using the interval-notation parameters, such as the *m* and *n* parameters with the **ed** command.
SCHAR_MAX	Maximum value of a type **signed char**.
SCHAR_MIN	Minimum value of a type **signed char**.
SHRT_MAX	Maximum value of a type **short**.
SHRT_MIN	Minimum value of a type **short**.
SSIZE_MAX	Maximum value of an object of type **ssize_t**.
STREAM_MAX	Number of streams that one process can have open at one time.
TMP_MAX	Minimum number of unique path names generated by the **tmpnam** subroutine. Maximum number of times an application can reliably call the **tmpnam** subroutine.
TZNAME_MAX	Maximum number of bytes supported for the name of a time zone (not the length of the **TZ** environment variable).
UCHAR_MAX	Maximum value of a type **unsigned char**.
UINT_MAX	Maximum value of a type **unsigned int**.
ULONG_MAX	Maximum value of a type **unsigned long int**.
USHRT_MAX	Maximum value of a type **unsigned short int**.
WORD_BIT	Number of bits in a word or type **int**.
KERNEL_BITMODE	Bit mode of the kernel, 32-bit or 64-bit.
REAL_MEMORY	Real memory size.
HARDWARE_BITMODE	Bit mode of the machine hardware, 32-bit or 64-bit.
MP_CAPABLE	MP-capability of the machine.

System Standards Configuration Variables

System standards configuration variables contain the *minimum* values required by a particular system standard. The **_POSIX_**, **POSIX2_**, and **_XOPEN_** prefixes indicate that the variable contains the minimum value for a system characteristic required by the POSIX 1003.1, POSIX 1003.2, and X/Open system standards, respectively. System standards are systemwide minimums that the system meets to support the particular system standard. Actual Configuration values may exceed these standards. The system standards configuration variables for the **getconf** command are defined as follows:

_POSIX_ARG_MAX	Maximum length, in bytes, of the arguments for one of the **exec** subroutines, including environment data.
_POSIX_CHILD_MAX	Maximum number of simultaneous processes for each real user ID.
_POSIX_JOB_CONTROL	Value of 1 if the system supports job control.
_POSIX_LINK_MAX	Maximum number of links to a single file.
_POSIX_MAX_CANON	Maximum number of bytes in a terminal canonical input queue.
_POSIX_MAX_INPUT	Maximum number of bytes allowed in a terminal input queue.
_POSIX_NAME_MAX	Maximum number of bytes in a file name (not including terminating null).
_POSIX_NGROUPS_MAX	Maximum number of simultaneous supplementary group IDs for each process.
_POSIX_OPEN_MAX	Maximum number of files that one process can have open at one time.
_POSIX_PATH_MAX	Maximum number of bytes in a path name.
_POSIX_PIPE_BUF	Maximum number of bytes guaranteed to be atomic when writing to a pipe.
_POSIX_SAVED_IDS	Value of 1. Each process has a saved set-user-ID and a saved set-group-ID.

_POSIX_SSIZE_MAX	Maximum value that can be stored in an object of type **ssize_t**.
_POSIX_STREAM_MAX	Number of streams that one process can have open at one time.
_POSIX_TZNAME_MAX	Maximum number of bytes supported for the name of a time zone (not the length of the **TZ** environment variable).
_POSIX_VERSION	Version of the POSIX 1 standard (C Language Binding) to which the operating system conforms.
_XOPEN_CRYPT	Value of 1 if the system supports the X/Open Encryption Feature Group.
_XOPEN_ENH_I18N	Value of 1 if the system supports the X/Open Enhanced Internationalisation Feature Group.
_XOPEN_SHM	Value of 1 if the system supports the X/Open Shared Memory Feature Group.
_XOPEN_VERSION	Version of the X/Open Portability Guide to which the operating system conforms.
_XOPEN_XCU_VERSION	Version of the X/Open Commands and Utilities specification to which the operating system conforms.
_XOPEN_XPG2	Value of 1 if the system supports the X/Open Portability Guide, Volume 2, January 1987, XVS System Calls and Libraries, otherwise undefined.
_XOPEN_XPG3	Value of 1 if the system supports the X/Open Specification, February 1992, System Interfaces and Headers, Issue 3, otherwise undefined.
_XOPEN_XPG4	Value of 1 if the system supports the X/Open CAE Specification, July 1992, System Interfaces and Headers, Issue 4, otherwise undefined.
POSIX2_BC_BASE_MAX	Maximum value allowed for the **obase** variable with the **bc** command.
POSIX2_BC_DIM_MAX	Maximum number of elements permitted in an array by the **bc** command.
POSIX2_BC_SCALE_MAX	Maximum value allowed for the **scale** variable with the **bc** command.
POSIX2_BC_STRING_MAX	Maximum length of a string constant accepted by the **bc** command.
POSIX2_CHAR_TERM	Value of 1 if the system supports at least one terminal type; otherwise it has the value -1.
POSIX2_COLL_WEIGHTS_MAX	Maximum number of weights that can be assigned to an entry of the **LC_COLLATE** locale variable in a locale-definition file.
POSIX2_C_BIND	Value of 1 if the system supports the C Language Binding Option from POSIX 2; otherwise, it has the value -1.
POSIX2_C_DEV	Value of 1 if the system supports the C Language Development Utilities from POSIX 2; otherwise, it has the value -1.
POSIX2_C_VERSION	Version of the POSIX 2 standard (C Language Binding) to which the operating system conforms.
POSIX2_EXPR_NEST_MAX	Maximum number of expressions that can be nested within parentheses by the **expr** command.
POSIX2_FORT_DEV	Value of 1 if the system supports the FORTRAN Development Utilities Option from POSIX 2; otherwise, it has the value -1.
POSIX2_FORT_RUN	Value of 1 if the system supports the FORTRAN Runtime Utilities Option from POSIX 2; otherwise, it has the value -1.
POSIX2_LINE_MAX	The maximum length, in bytes, of a command's input line (either standard input or another file) when the command is described as processing text files. The length includes room for the trailing new-line character.
POSIX2_LOCALEDEF	Value of 1 if the system supports the creation of the locales by the **localedef** command; otherwise, it is undefined.
POSIX2_RE_DUP_MAX	Maximum number of repeated occurrences of a regular expression permitted when using the interval-notation parameters, such as the *m* and *n* parameters with the **ed** command.
POSIX2_SW_DEV	Value of 1 if the system supports the Software Development Utilities Option; otherwise, it has the value -1.
POSIX2_UPE	Value of 1 if the system supports the User Portability Utilities Option from POSIX 2; otherwise, it as the value -1.

POSIX2_VERSION	Date of approval of the most current version of the POSIX 2 standard that the system supports. The date is a six-digit number, with the first four digits signifying the year and the last two digits the month. Different versions of the POSIX 2 standard are periodically approved by the IEEE Standards Board, and the date of approval is used to distinguish between different versions.

System Path Configuration Variables

The *PathConfiguration* parameter specifies system path configuration variables whose values contain information about paths and path structures in the system. The following list defines these variables:

_POSIX_CHOWN_RESTRICTED	The **chown**() subroutine is restricted to a process with appropriate privileges, and to changing the group ID of a file only to the effective group ID of the process or to one of its supplementary group IDs. If the *PathName* parameter refers to a directory, the value returned applies to any files except directories that exist or can be created within the directory.
_POSIX_NO_TRUNC	Path names longer than the limit specified by the *NAME_MAX* variable will generate an error. If the *PathName* parameter refers to a directory, the value returned applies to file names within the directory.
_POSIX_VDISABLE	Terminal special characters, defined in the **termios.h** file, can be disabled using this character value.
LINK_MAX	Maximum number of links to a single file. If the *PathName* parameter refers to a directory, the value returned applies to the directory.
MAX_CANON	Maximum number of bytes in a terminal canonical input line.
MAX_INPUT	Maximum number of bytes for which space is available in a terminal input queue.
NAME_MAX	Maximum number of bytes in a file name (not including terminating null). If the *PathName* parameter refers to a directory, the value returned applies to the file names within the directory.
PATH_MAX	Maximum number of bytes in a path name, including the terminating null character. If the *PathName* parameter refers to a directory, the value returned is the maximum length of a relative path name when the specified directory is the working directory.
PIPE_BUF	Maximum number of bytes guaranteed to be atomic when writing to a pipe. If the *PathName* parameter refers to a FIFO or a pipe, the value returned applies to the referenced object. If the *PathName* parameter refers to a directory, the value returned applies to any FIFO that exists or can be created within the directory.
DISK_PARTITION	Physical partition size of the disk. **Note:** For the DISK_PARTITION path configuration variable, the *PathName* parameter must specify the complete path of the disk for which information is being queried.
DISK_SIZE	Disk size in megabytes. **Note:** For the **DISK_SIZE** path configuration variable, the *PathName* parameter must specify the complete path of the disk for which information is being queried.

Device Variables

The *DeviceVariable* parameter indicates that the *DeviceName* parameter is the path of a device, such as **/dev/hdisk0**. Given the path of a disk, the **getconf** command displays the device name or location of the disk.

DISK_DEVNAME	Device name or location of the device.

Exit Status

This command returns the following exit values:

0 The specified variable is valid and information about its current state was successfully written.
>0 An error occurred.

1. To display the value of the **ARG_MAX** variable, type:

   ```
   getconf ARG_MAX
   ```

2. To display the values of all system configuration variables, type:

   ```
   getconf -a
   ```

3. To display the value of the **NAME_MAX** variable for the **/usr** directory, type:

   ```
   getconf NAME_MAX /usr
   ```

4. The following sequence of shell commands shows how to handle unspecified results:

   ```
   if value=$(getconf PATH_MAX /usr)
   then
        if [ "$value" = "undefined" ]
        then
                   echo
                        The value of PATH_MAX in /usr is undefined.
        else

                        The value of PATH_MAX in /usr is $value.
        fi
   else
                   echo
                        Error in the getconf command.
   fi
   ```

Examples

1. To display the value of the **ARG_MAX** variable, type:

   ```
   getconf ARG_MAX
   ```

2. To display the value of the **NAME_MAX** variable for the **/usr** directory, type:

   ```
   getconf NAME_MAX /usr
   ```

3. The following sequence of shell commands shows how to handle unspecified results:

   ```
   if value=$(getconf PATH_MAX /usr)
   then
        if [ "$value" = "undefined" ]
        then
                echo
                      The value of PATH_MAX in /usr is undefined.
        else
                echo
                      The value of PATH_MAX in /usr is $value.
        fi
   else
        echo Error in the getconf command.
   fi
   ```

4. If the command:

   ```
   getconf _XBS5_ILP32_OFF32
   ```

   does not write `-1\n` or `undefined\n` to standard output, then commands of the form:

   ```
   getconf -v XBS5_ILP32_OFF32 ...
   ```

   will determine values for configuration variables corresponding to the `XBS5_ILP32_OFF32` compilation environment specified in **c89**, Extended Description.

5. If the command:

`getconf _XBS5_ILP32_OFFBIG`

does not write `-1\n` or `undefined\n` to standard output, then commands of the form:

`getconf -v XBS5_ILP32_OFFBIG ...`

will determine values for configuration variables corresponding to the `XBS5_ILP32_OFFBIG` compilation environment specified in **c89**, Extended Description.

6. If the command:

   `getconf _XBS5_LP64_OFF64`

   does not write `-1\n` or `undefined\n″` to standard output, then commands of the | form:

   `getconf -v XBS5_LP64_OFF64 ...`

   will determine values for configuration variables corresponding to the XBS5_LP64_OFF64 compilation environment specified in c89, Extended Description.

7. If the command:

   `getconf _XBS5_LPBIG_OFFBIG`

   does not write `-1\n` or `undefined\n` to standard output, then commands of the form:

   `getconf -v _XBS5_LPBIG_OFFBIG`

   will determine values for configuration variables corresponding to the XBS5_LPBIG_OFFBIG compilation environment specified in c89, Extended Description.

8. To determine the disk size for disk `hdisk0`, as root user, enter the following:

   `getconf DISK_SIZE /dev/hdisk0`

9. To determine the real memory size, enter the following:

   `getconf REAL_MEMORY`

10. To determine if the machine hardware is 32-bit or 64-bit, enter the following:

    `getconf HARDWARE_BITMODE`

11. To determine if the kernel is 32-bit or 64-bit, enter the following:

    `getconf KERNEL_BITMODE`

12. To determine the device name or location of the disk `hdisk0`, enter the following:

    `getconf DEVICE_NAME /dev/hdisk0`

Files

/usr/bin/getconf	Contains the **getconf** command.
/usr/include/limits.h	Defines system configuration variables.
/usr/include/unistd.h	Defines system configuration variables.

Related Information

The **confstr** subroutine, **pathconf** subroutine, **sysconf** subroutine.

Commands in *Operating system and device management*.

---

getdev Command

Purpose

Lists devices that match the specified criteria.

Syntax

getdev [**-a**] [**-e**] [*Criteria*] [*DeviceList*]

Description

Lists devices that match the given criteria. The criteria is given in the form of expressions. The **getdev** command can check all devices on the system or a specified list of devices.

Flags

-a Specifies that a device must match all criteria to be included in the list generated by this command. The **-a** flag has no effect if no criteria are defined.

-e Specifies that the devices provided in the *devicelist* be excluded from the list generated by the **getdev** command. Without the *-e* flag only devices in the *devicelist* are generated. This flag is ignored if no devices are specified.

Parameters

Criteria Defines criteria that a device must match before it can be included in the generated list. *Criteria* can be specified as an expression or a list of expressions which a device must meet for it to be included in the list generated by **getdev**. If no criteria are provided, all devices are included in the list.

Devices must satisfy at least one of the criteria in the list. However, the **-a** option can be used to specify that a ″logical and″ operation should be performed. Then, only those devices that match all of the criteria in a list will be included.

There are four possible expression types which the criteria specified in the *Criteria* parameter may follow:

*Attribute***=***Value*
Fetches all devices with a member which has*Attribute* defined and is equal to *Value*.

*Attribute***!=***Value*
Fetches all devices with a member which has *Attribute* defined and does not equal *Value*.

*Attribute***:***
Fetches all devices with a member which has *Attribute* defined.

*Attribute***!:***
Fetches all devices with a member which does not have *Attribute* defined.

The following are the valid device attributes:

alias The name by which a device is known.

desc A description of the device.

type A token describing the type of the device.
The valid set of values for the **type** attribute can be obtained by executing the following command. **odmget PdDv | grep -w class | awk '{print $3}' | sed 's/"//g' | sort | uniq**

status The current state of the device.
The list of possible values for status are:
1.Defined
2.Available
3.Stopped
4.Diagnose
The values for **status** are not case sensitive.

DeviceList Specifies a space-separated list of devices to be checked for the *Criteria*.

Exit Status

0 The command completed successfully

> 1 Failure has occurred.

Examples

1. To display all devices, enter:
   ```
   getdev
   ```
2. To list devices which are of type ″logical_volume″, enter:
   ```
   getdev type=logical_volume
   ```
3. To list devices which are not of type ″logical_volume″, enter:
   ```
   getdev type!=logical_volume
   ```
4. To list devices which are of type ″logical_volume″ or whose device alias is ″sys0″, enter:
   ```
   getdev type=logical_volume alias=sys0
   ```

   The output will look similar to the following:
   ```
   hd1
   hd2
   hd3
   hd4
   ...
   sys0
   ```
5. To list devices which are of type ″logical_volume″ and whose device alias is ″lv01″, enter:
   ```
   getdev -a type=logical_volume alias=lv01
   ```
6. To display devices for which the **status** attribute is defined , enter:
   ```
   getdev status:*
   ```
7. To display devices for which the **desc** attribute is not defined , enter:
   ```
   getdev desc!:*
   ```

Files

/usr/sbin/getdev	Contains the **getdev** command

Related Information

The **getdgrp** command, **lsdev** command.

---

getdgrp Command

Purpose

Lists device classes that match the specified criteria.

Syntax

getdgrp [**-a**] [**-e**] [**-l**][*Criteria*] [*DeviceClassList*]

Description

Lists device classes that contain devices matching the given criteria. The criteria is given in the form of expressions.

Flags

-a Indicates that a device must match all criteria of the device class to be included in the report generated by this command. The **-a** flag has no effect if no criteria are defined.

-e Indicates that thedevices classes specified in the parameter list be excluded from the report generated by this command. The **-e** flag has no effect if no devices are specified.

-l Indicates that all device classes that are subject to the **-e** option and the **dgroup** list, be listed even if they contain no valid device members. This option has no affect if *Criteria* is specified on the command line.

Parameters

Criteria Defines criteria that a device must match before a device class to which it belongs can be included in the generated list. *Criteria* can be specified as an expression or a list of expressions which a device must meet for its class to be included in the list generated by **getdgrp**. If no criteria are given, all device classes are included in the list.

Devices must satisfy at least one of the criteria in the list. However, the **-a** option can be used to specify that a ″logical and″ operation should be performed. Then, only those classes containing devices that match all of the criteria in a list will be included.

There are four possible expression types which the criteria specified in the *Criteria* parameter may follow:

*Attribute***=***Value*
Fetches all device classes with a member which has*Attribute* defined and is equal to *Value*.

*Attribute***!=***Value*
Fetches all device classes with a member which has *Attribute* defined and does not equal *Value*.

*Attribute***:***
Fetches all device classes with a member which has *Attribute* defined.

*Attribute***!:***
Fetches all device classes with a member which does not have *Attribute* defined.

The following are the valid device attributes:

alias The name by which a device is known.

desc A description of the device.

type A token describing the type of the device.

status The current state of the device.
The list of possible values for status are :
1.Defined
2.Available
3.Stopped
4.Diagnose
The values for **status** are not case sensitive.

DeviceClassList Specifies device class name in the Customized Device Configuration database or in the Predefined Device Configuration database.

Exit Status

0 The command completed successfully

1 Command syntax was incorrect, invalid option was used, or an internal error occurred.

2 The Customized Devices object class or the Predefined Devices object class could not be opened for reading.

Examples

1. To display all device classes, enter:

   ```
   getdgrp
   ```

   The output looks similar to the following:

   ```
   adapter
   aio
   bus
   cdrom
   disk
   diskette
   gxme
   if
   keyboard
   lft
   logical_volume
   lvm
   memory
   mouse
   planar
   processor
   pty
   pwrmgt
   rcm
   sys
   tape
   tcpip
   tty
   ```

2. To list device classes whose devices are of type ″logical_volume″, enter:

   ```
   getdgrp type=logical_volume
   ```

   The output looks like the following:

   ```
   logical_volume
   ```

3. To list device classes whose devices are of type ″logical_volume″ or whose device alias is ″sys0″, enter:

   ```
   getdgrp type=logical_volume alias=sys0
   ```

   The output looks like the following:

   ```
   logical_volume
   sys
   ```

4. To list device classes whose devices status attribute is defined , enter:

   ```
   getdgrp status=defined
   ```

   The output looks like the following:

   ```
   logical_volume
   posix_aio
   rcm
   ```

5. To display device classes for whose devices the **status** attribute is defined and belong to the ″processor″ device class, enter:

   ```
   getdgrp status:* processor
   ```

   The output looks like the following:

   ```
   processor
   ```

6. To display device classes for whose devices the **status** attribute is not defined, enter:

```
getdgrp status!:* processor
```

Files

/usr/sbin/getdgrp	Contains the **getdgrp** command

Related Information

The **getdev** command, **lsdev** command.

---

getea Command

Purpose

Retrieves named extended attributes from a file.

Syntax

getea [**-n** *Name*] [**-l**] [**-e** *RegExp*] [**-s**] *FileName*

Description

The **getea** command reads named extended attributes from a file. If the **-n** *Name* parameter is specified then just extended attributes matching *Name* are retrieved.

Note: To prevent naming collisions, JFS2 has reserved the 8-character prefix `(0xf8)SYSTEM(0xF8)` for system-defined extended attributes. Avoid using this prefix for naming user-defined extended attributes.

If the **-e** *RegExp* parameter is specified then just extended attributes matching the regular expression *RegExp* are retrieved. If neither **-n** or **-e** flags are specified all extended attributes are retrieved.

This command is not used to get ACLs. The **aclget** command is used to get ACLs.

Flags

-e *RegExp*	Specifies a regular expression to retrieve all extended attributes which match. The values are displayed in character format.
-l	Specifies to get the extended attributes from the symbolic link itself rather than the file to which it is pointing.
-n *Name*	Specifies the name of specific extended attributes to retrieve. The values are displayed in character format.
-s	Displays only the names and not the values for the extended attributes.
FileName	Specifies the file from which to read the extended attributes.

Exit Status

0	Successful completion.
Positive integer	An error occurred.

Security

Attention RBAC users and Trusted AIX users: This command can perform privileged operations. Only privileged users can run privileged operations. For more information about authorizations and privileges,

see Privileged Command Database in *Security*. For a list of privileges and the authorizations associated with this command, see the **/etc/security/privcmds** database file.

Examples

1. To retrieve all named extended attributes for the file `design.html`, type:
   ```
   getea design.html
   ```
2. To retrieve the named extended attribute, `Approver`, for the file `design.html`, type:
   ```
   getea -n Approver design.html
   ```
3. To retrieve just the names of all named extended attributes for the file `design.html`, type:
   ```
   getea -s design.html
   ```
4. To retrieve all named extended attributes for the symbolic link `design.html`, type:
   ```
   getea -l design.html
   ```

Location

/usr/sbin

Related Information

chfs, **crfs** commands in *AIX Version 6.1 Commands Reference, Volume 1*.

setea command in *AIX Version 6.1 Commands Reference, Volume 5*.

---

getopt Command

Purpose

Parses command line flags and parameters.

Syntax

getopt *Format Tokens*

Description

The **getopt** command parses a list of tokens using a format that specifies expected flags and arguments. A flag is a single ASCII letter and when followed by a : (colon) is expected to have an argument that may or may not be separated from it by one or more tabs or spaces. You can include multibyte characters in arguments, but not as a flag letter.

The **getopt** command completes processing when it has read all tokens or when it encounters the special token — (double hyphen). The **getopt** command then outputs the processed flags, a — (double hyphen), and any remaining tokens.

If a token fails to match a flag, the **getopt** command writes a message to standard error.

Examples

The **getopt** command can be used in a skeleton shell script to parse options, as in the following example:

```
#!/usr/bin/bsh
# parse command line into arguments
set -- `getopt a:bc $*`
# check result of parsing
if [ $? != 0 ]
then
        exit 1
fi
while [ $1 != -- ]
```

```
do
        case $1 in
        -a)     # set up the -a flag
                AFLG=1
                AARG=$2
                shift;;
        -b)     # set up the -b flag
                BFLG=1;;
        -c)     # set up the -c flag
                CFLG=1;;
        esac
        shift   # next flag
done
shift   # skip --
# now do the work
.
.
.
```

Note: In the C shell, use the following command to run the **getopt** command:

```
set argv=`getopt OptionString $*`
```

In each of the following examples, the **getopt** command would process the flags and arguments in the same way:

- `-a ARG -b -c`
- `-a ARG -bc`
- `-aARG -b -c`
- `-b -c -a ARG`

Files

/usr/bin/getopt	Contains the **getopt** command.

Related Information

The **bsh** command, **csh** command.

The **getopt** subroutine.

Shells in *Operating system and device management*.

---

getopts Command

Purpose

Processes command-line arguments and checks for valid options.

Syntax

getopts *OptionString Name* [*Argument* ...]

Description

The **getopts** command is a Korn/POSIX Shell built-in command that retrieves options and option-arguments from a list of parameters. An option begins with a **+** (plus sign) or a **-** (minus sign) followed by a character. An option that does not begin with either a **+** or a **-** ends the *OptionString*. Each time the **getopts** command is invoked, it places the value of the next option in *Name* and the index of the

next argument to be processed in the shell variable **OPTIND**. Whenever the shell is invoked, **OPTIND** is intialized to 1. When an option begins with **+**, a **+** is prepended to the value in *Name*.

If a character in *OptionString* is followed by a **:** (colon), that option is expected to have an argument. When an option requires an option-argument, the **getopts** command places it in the variable **OPTARG**.

When an option character not contained in *OptionString* is found, or an option found does not have the required option-argument:

- If *OptionString* does not begin with a **:** (colon),
  - *Name* will be set to a **?** (question mark) character,
  - **OPTARG**. will be unset, and
  - a diagnostic message will be written to standard error.

This condition is considered to be an error detected in the way arguments were presented to the invoking application, but is not an error in the processing of the **getopts** command; a diagnostic message will be written as stated, but the exit status will be zero.

- If *OptionString* begins with a **:** (colon),
  - *Name* will be set to a **?** (question mark) character for an unknown option or to a **:** (colon) character for a missing required option,
  - **OPTARG** will be set to the option character found, and
  - no output will be written to standard error.

Any of the following identifies the end of options: the special option **- -,** finding an argument that does not begin with a **-**, or **+**, or encountering an error.

When the end of options is encountered:

- the **getopts** command will exit with a return value greater than zero,
- **OPTARG** will be set to the index of the first non-option-argument, where the first **- -** argument is considered to be an option-argument if there are no other non-option-arguments appearing before it, or the value $#+1 if there are no non-option-arguments,
- *Name* will be set to a **?** (question mark) character.

Parameters

OptionString — Contains the string of option characters recognized by the **getopts** command. If a character is followed by a colon, that option is expected to have an argument, which should be supplied as a separate argument. The options can be separated from the argument by blanks. The first character in *OptionString* determines how the **getopts** command behaves if an option character is not known or an option-argument is missing.

> **Note:** The characters question mark and colon must not be used as option characters by an apllication. The use of other characters that are not alphanumeric produces unspecified results.

Name — Set by the **getopts** command to the option character that was found.

Argument ... — One or more strings separated by white space, checked by the **getopts** command for legal options. If *Argument* is omitted, the positional parameters are used. See Parameter substitution in the Korn shell or POSIX shell in the Korn Shell for more information on positional parameters.

> **Note:** Generally, you won't specify *Argument* as part of the **getopts** command, but it may be helpful when debugging your script.

Exit Status

This command returns the following exit values:

0 An option, specified or unspecified by *OptionString*, was found.
>0 The end of options was encountered or an error occurred.

Examples

1. The following **getopts** command specifies that `a`, `b`, and `c` are valid options, and that options `a` and `c` have arguments:

   ```
   getopts a:bc: OPT
   ```

2. The following **getopts** command specifies that `a`, `b`, and `c` are valid options, that options `a` and `b` have arguments, and that **getopts** set the value of `OPT` to **?** when it encounters an undefined option on the command line:

   ```
   getopts :a:b:c OPT
   ```

3. The following script parses and displays it arguments:

   ```
   aflag=
   bflag=

   while getopts ab: name
   do
               case $name in
               a)     aflag=1;;
               b)     bflag=1
                             bval="$OPTARG";;
               ?)     printf "Usage: %s: [-a] [-b value] args\n" $0
                             exit 2;;
              esac
   done

   if [ ! -z "$aflag" ]; then
              printf "Option -a specified\n"
   fi

   if [ ! -z "$bflag" ]; then
              printf 'Option -b "%s" specified\n' "$bval"
   fi

   shift $(($OPTIND -1))
   printf "Remaining arguments are: %s\n" "$*"
   ```

Related Information

Korn shell or POSIX shell commands in *Operating system and device management*.

---

getrunmode Command

Purpose

Displays the mode the system is running in.

Syntax

getrunmode

Description

The **getrunmode** command displays the mode the system is running in. A run mode is either the CONFIGURATION mode or the OPERATIONAL mode.

Examples

To retrieve the run mode, enter:

```
getrunmode
```

Files

/usr/sbin/getrunmode	Contains the **getrunmode** command.

Related Information

The **setrunmode** command.

Trusted AIX in *Security*.

---

getsecconf Command

Purpose

Displays the system security flags.

Syntax

getsecconf { **-c** | **-o** }

Description

The **getsecconf** command displays the system security flags. When invoked without any options, the **getsecconf** command displays the security flags that pertain to the mode the system is running in.

Flags

-c	Specifies the CONFIGURATION mode.
-o	Specifies the OPERATIONAL mode.

Security

The **getsecconf** command is a privileged command. Successfully running this command requires the following authorization:

aix.mls.system.config.read	Required to list the system security flags.

Exit Status

The **getsecconf** command returns the following exit values:

0	Successful execution.
>0	An error occurred.

Examples

1. To display the system security flags in the CONFIGURATION mode, enter:

   ```
   getsecconf –c
   ```

2. To display the system security flags in the OPERATIONAL mode, enter:

   ```
   getsecconf –o
   ```

Files

/usr/sbin/getsecconf	Contains the **getsecconf** command.

Related Information

The **setsecconf** command.

Trusted AIX in *Security*.

---

getsyslab Command

Purpose

Displays the minimum and maximum labels of the system.

Syntax

getsyslab

Description

The **getsyslab** command is used to display the system minimum and maximum sensitivity label (SL), and minimum and maximum integrity label (TL).

Security

The **getsyslab** command is a privileged command. Successfully running the command requires the following authorization:

aix.mls.system.label.read	Required to list the system labels.

Files Accessed:

Mode	File
r	/etc/security/enc/LabelEncodings

Example

To display the system labels, enter:

```
getsyslab
```

Files

/usr/sbin/getsyslab	Contains the **getsyslab** command.
/etc/security/enc/ LabelEncodings	System default label encodings file.

Related Information

The **setsyslab** command.

Trusted AIX in *Security*.

gettable Command

Purpose
Gets Network Information Center (NIC) format host tables from a host.

Syntax
/usr/sbin/gettable [**-v**] *Host* [*OutFile*]

Description
The **/usr/sbin/gettable** command is used to obtain the NIC standard host tables from a server indicated by the *Host* parameter. The tables, if retrieved, are placed in the file indicated by the *OutFile* parameter.

The **gettable** command opens a Transmission Control Protocol (TCP) connection to the port indicated in the service specification for the *Host* parameter. A request is then made for all names, and the resultant information is placed in the output file.

The **gettable** command is best used in conjunction with the **htable** command, which converts the NIC standard file format to that used by the network library lookup routines.

Flags
-v Gets just the version number instead of the complete host table and puts the output in *OutFile* or, by default, in a file named **hosts.ver**.

Parameters
Host Specifies the server that provides the host table information.
OutFile Specifies the file where you want to place the host table information. If you use the **gettable** command without the **-v** flag, the default file name is **hosts.txt**.

Related Information
The **htable** command.

Transmission Control Protocol (TCP) and TCP/IP protocols in *Networks and communication management*.

gettrc Command

Purpose
Manages the collection of trace files.

Syntax
gettrc [**-c**] [**-C** *dirname*] [**-m**] [**-M** *dirname*] [**-s**] [**-S** *dirname*]

Description
The **gettrc** command is a script that is used in conjunction with the **snap** command. It manages the collection of system trace files, lightweight memory trace (LMT) files, and component trace (CT) files.

Flags

-c	Collects the component trace files.
-C *dirname*	Collects component trace files from the directory specified by *dirname*.
-m	Collects memory trace files.
-M *dirname*	Collects lightweight memory trace files from the directory specified by *dirname*
-s	Collects system trace files.
-S *dirname*	Collects system trace files from the directory specified by *dirname*

Exit Status

0	The command completed successfully.
>0	An error occurred.

Examples

1. To use **gettrc** in conjunction with the **snap** command to retrieve the different kinds of trace files, enter:

   ```
   snap "gettrc -c -C dirname -m -M dirname -s -S dirname"
   ```

   This will return system trace files, LMT files, and CT files, including those listed in the directory specified by *dirname*.

Location

/usr/lib/ras/snapscripts/gettrc

Files

/usr/lib/ras/cpufmt
/etc/trcfmt

Related Information

The **snap** command.

---

getty Command

Purpose

Sets the characteristics of ports.

Syntax

getty [[**-r** | **-u** | **-U**] [**-d**] [**-H** *HeraldString*] [**-M** *motdFile*] [**-N**]] *PortName*

Description

The **getty** command sets and manages terminal lines and ports. The **getty** command is run by the **init** command. The **getty** command is linked to the Terminal State Manager program. The Terminal State Manager program provides combined terminal control and login functions.

Note: The **getty** command is not entered on the command line.

When invoked as the **getty** command, the Terminal State Manager program provides the normal port management functions that include:

Bidirectional use	Allows terminal lines to be used to initiate and accept connections.
Line speed	Sets the baud rates for sending and receiving.
Parity	Sets the parity to be even, odd or none.
Delays	Sets the delays for carriage return, tab, new line, and form feed.
Character set mapping	Sets the character set mapping for case, tabs, and carriage control.
Logger Program	Specifies the program used to log the user into the system. If the attribute is set, the Secure Attention Key (SAK) processing is disabled. If the attribute is not set, it defaults to **/usr/sbin/login**. The **logger** attribute is contained within the Object Data Manager (ODM) database.
Character and line erase	Sets the keystroke used for character and line erase.
Echoing mode	Sets the echo to local or remote.

When the **getty** command is invoked, the following steps occur:

1. The port protection is set according to the **owner** and **protection** attributes in the ODM database. If these attributes are not specified, they default to root and 622.
2. The port specified by the *PortName* parameter is opened. If the carrier detection is available on the port, the open does not complete until the carrier is present or another process has lost the carrier with the port.
3. The specified port might be locked. If the **getty** command is run with the **-u** or **-r** flag, it attempts to lock the port. If the port is already locked the command waits until the port is available and then exits. If the **-r** flag was specified, the **getty** command waits for a byte of data to be received on the port before continuing.
4. The terminal attributes are set according to the configuration information for the specified port. Secure Attention Key processing can be enabled at this point depending on the system configuration.
5. The herald message is written to the specified port.
6. The login name is read from the specified port. If a framing error or a break occurs, the **getty** command repeats steps four and five with the next group of configured terminal attributes. This is most commonly used to cycle the baud rates for modems. But any ODM field (except `logmodes` and `runmodes`) may be cycled by entering a list of comma separated values in the ODM database.
7. The terminal modes are reset according to the `runmodes` parameter and the login name. If the login name is terminated by a new line, the **getty** command turns on the carriage-return to new line mapping. If all alphabetic characters are in uppercase, the user is prompted to log in using lowercase characters if possible, and mapping from lowercase to uppercase is turned on.
8. If a program is specified by the logger parameter, it is executed and Secure Attention Key processing is disabled. Otherwise, the Terminal State Manager program performs a standard system login.

> **Note:** If the Secure Attention Key sequence is typed during a user login, the user is logged into the trusted shell (if the system is configured where that port is trusted and the user is allowed on the trusted path).

Flags

-d	Provides debugging information.
-H *HeraldString*	Specifies an alternate herald message to write on the port to prompt for a login name. The message string must be one word and cannot contain any spaces. This string will take precedence over herald messages defined in the **/etc/security/login.cfg** file. If no string is specified with this option or in the **login.cfg** file, the default herald from the message catalog will be used.
-M *motdFile*	Specifies the path to an alternate message of the day file. If not specified, this value is **/etc/motd** by default.

-N	Causes **getty** to bypass any checking for the process ID in the **/etc/utmp** file. This allows a process other than the lowest login shell to **exec getty**.
-r	Makes the port available for shared (bi-directional) use. If the lock is unsuccessful, the **getty** command waits until the lock is available and then exits. If the lock is successful, the **getty** command waits for a byte of data on the port after locking the port.
-u	Makes the port available for shared (bi-directional) use. If the lock is unsuccessful, the **getty** command waits until the lock is available and then exits.
-U	Same as the **-u** flag, except **getty** does not wait for the lock to be available. It makes the port available regardless of the lock.

Security

Access Control: This program should be installed as a program in the Trusted Computing Base, executable by any user and **setuid** to root.

Example

To enable logging onto tty0, add the following line to the **/etc/inittab** file:

```
tty0:2:respawn: /usr/sbin/getty /dev/tty0
```

This command initializes the port `/dev/tty0` and sets up the characteristics of the port.

Files

/usr/sbin/getty	Contains the **getty** command.
/etc/locks	Contains lock files that prevent multiple uses of communications devices and multiple calls to remote systems.
/usr/sbin/login	The **login** command.
/etc/security/login.cfg	Contains port login configurations.
/etc/motd	Contains the message of the day displayed after login.
/usr/bin/setmaps	The **setmaps** command.
/etc/utmp	Contains information about users logged into the system.

Related Information

The **login** command, **setgroups** command, **shell** command, **su** command, **telinit or init** command, **tsm** command.

Object Data Manager (ODM) Overview for Programmers in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

---

glbd Daemon

Purpose

Manages the global location broker database.

Syntax

/etc/ncs/glbd [**-create** { **-first** [**-family** *FamilyName*] | **-from** *HostName* }] [**-change_family** *FamilyName*] [**-listen** *FamilyList*] [**-version**]

Description

The **glbd** daemon manages the global location broker (GLB) database. The GLB database, part of the Network Computing System (NCS), helps clients to clients to locate servers on a network or internet. The

GLB database stores the locations (specifically, the network addresses and port numbers) of servers on which processes are running. The **glbd** daemon maintains this database and provides access to it.

There are two versions of the GLB daemon, **glbd** and **nrglbd**.

You can replicate the GLB database to increase its availability. Copies of the database can exist on several hosts, with a **glbd** running on each of those hosts to maintain the consistency of the database replicas. (In an internet, at least one **glbd** must be running in each network.) Each replica of the GLB keeps a list of all the other GLB replicas. The **drm_admin** tool administers the replication of the GLB database and of the replica list.

Currently, **glbd** supports both the DARPA IP and Domain DDS network protocols. A GLB replica can allow access to its database from both IP and DDS clients. However, when communicating with each other to maintain replication of the GLB database, GLB replicas should use only one protocol family. You choose which family the GLBs will use. In an internet, all routing nodes must support this family.

The **glbd** daemon can be started in one of two ways:

- Through the System Resource Controller (the recommended method), by entering on the command line:
  ```
  startsrc -s glbd
  ```
- By a person with root user authority entering on the command line:
  ```
  /etc/ncs/glbd &
  ```

TCP/IP must be configured and running on your system before starting the **glbd** daemon. The **llbd** daemon must also be started and running before you start the **glbd** daemon.

Flags

-create Creates a replica of the GLB. This option creates a GLB database in addition to starting a broker process. It must be used with either -first or -from.

-first Creates the first replica (that is, the very first instance) of the GLB on your network or internet. This option can be used only with the **-create** option.

-family *FamilyName*
Specifies the address family that the first GLB replica will use to identify itself on the replica list. This option can be used only in conjunction with the **-first** option. Any subsequently created replicas must use this family to communicate with this replica. Currently, *FamilyName* can be either **dds** or **ip**. If this option is not used, the replica will be identified on the replica list by its DDS address.

-from *HostName*
Creates additional replicas of the GLB. This option can be used only with the **-create** option. A replica of the GLB must exist at *HostName*. The database and replica list for the new replica are initialized from those at *HostName*. The replica at *HostName* adds an entry for the new replica to its replica list and propagates the entry to the other GLB replicas.

A *HostName* takes the form family:host, where the host can be specified either by its name or by its network address. For example, `ip:jeeves`, `ip:bertie`, and `ip:#192.5.5.5` are acceptable host names.

The new replica will use the same address family as *HostName* in identifying itself on the replica list. For example, if *HostName* is an IP address, the new replica will be listed by its IP address on the replica list.

-change_family *FamilyName* Changes the address family of every GLB replica. Use this option only if network reconfigurations require that you make such a change. Currently, *FamilyName* can be either **dds** or **ip**.

-listen *FamilyList* Restricts the address families on which a GLB listens. Use it only if you are creating a special configuration where access to a GLB is restricted to a subset of hosts in the network or internet.

The *FamilyList* is a list of the address families on which the GLB will listen. Names in this list are separated by spaces. Possible family names include **dds** and **ip**.

The GLB will always listen for requests from the family by which it is listed on the replica list, even if that family is not specified in *FamilyList*.

If **glbd** is started without the **-listen** option, the GLB will listen on all address families that are supported both by NCS and by the local host. On Apollo systems, this set of families always includes **dds** and may also include **ip**. On most other systems, **ip** is currently the only family.

-version Displays the version of NCS that this **glbd** belongs to, but does not start the daemon.

Files

/etc/ncs/glb_log	Contains diagnostic output from **glbd**.
/etc/rc.ncs	Contains commands to start the NCS daemons.

Examples

1. Create and start for the first time the first replica of the GLB on this network or internet:
   ```
   /etc/ncs/glbd -create -first -family ip &
   ```
2. Start for the first time a subsequent replica of the GLB, initializing its database from host `jeeves`:
   ```
   /etc/ncs/glbd -create -from ip:jeeves &
   ```
3. Restart an existing replica of the GLB:
   ```
   /etc/ncs/glbd &
   ```

Related Information

The **drm_admin** command, **lb_admin** command, **startsrc** command.

The **llbd** daemon.

---

gprof Command

Purpose

Displays call graph profile data.

Syntax

/usr/ccs/bin/gprof [**-b**] [**-c** [*filename*]] [**-e** *Name*] [**-E** *Name*] [**-f** *Name*] [**-g** *filename*] [**-i** *filename*] [**-p** *filename*] [**-F** *Name*] [**-L** *PathName*] [**-s**] [**-x** [*filename*]] [**-z**] [**a.out** [**gmon.out** ...]]

Description

The **gprof** command produces an execution profile of C, Pascal, FORTRAN, or COBOL programs. The effect of called routines is incorporated into the profile of each caller. The **gprof** command is useful in identifying how a program consumes CPU resource. To find out which functions (routines) in the program are using the CPU, you can profile the program with the **gprof** command.

The profile data is taken from the call graph profile file (**gmon.out** by default) created by programs compiled with the **cc** command using the **-pg** option. The **-pg** option also links in versions of library routines compiled for profiling, and reads the symbol table in the named object file (**a.out** by default), correlating it with the call graph profile file. If more than one profile file is specified, the **gprof** command output shows the sum of the profile information in the given profile files.

The **-pg** option causes the compiler to insert a call to the **mcount** subroutine into the object code generated for each recompiled function of your program. During program execution, each time a parent calls a child function the child calls the **mcount** subroutine to increment a distinct counter for that parent-child pair. Programs not recompiled with the **-pg** option do not have the **mcount** subroutine inserted, and therefore keep no record of who called them.

Note: Symbols from C++ object files have their names demangled before they are used.

The GPROF environment variable can be used to set different options for profiling. The syntax of this environment variable is defined as follows:

```
GPROF = profile:<profile-type>,scale:<scaling-factor>,file:<file-type>,filename:<filename>
```

where:

- <profile-type> describes what type of profiling need to be performed, this can be either process or thread. Type 'process' indicates that profiling granularity is at process level, 'thread' indicates that profiling granularity is at thread level.
- <scaling-factor> describes how much memory need to be allocated for call graph profile, by default the scaling factor is 2 for process level profiling and 8 for thread level profiling. A scaling factor of 2 indicates that a memory of 1/2 of the process's size is allocated for every process or thread, scaling factor of 8 indicates that a memory of 1/8th of the process's size is allocated for every process of thread. This memory is the buffer area to store the call graph information.
- <file-type> describes what type of **gmon.out** file is required, a value of multi indicates that one gmon.out file per process is required, a value of multithread indicates that one **gmon.out** file per thread is required. If a application is profiled with -pg option and it does a fork then specifying multi generates one **gmon.out** file for the parent process and another for the child process. The naming convention for the generated **gmon.out** files are as follows:
  - For multi file-type: <prefix>-processname-pid.out
  - For multithread file-type: <prefix>-processname-pid-Pthread<threadid>.out

  The <prefix> is by default **gmon**. The user can define his own prefix by using the filename parameter of the GPROF environment variable.
- <filename> describes the prefix that need to be used for the generated **gmon.out** files. By default the prefix is **gmon**.

Note: Specifying `profile:thread` generates a format **gmon.out** file which can be read only by AIX 5.3 gprof. If a user wants a old-format **gmon.out** file and still want to specify **profile:thread**, then you must specify **file:multithread**, this will generate old format **gmon.out** file per thread. Hence if your application has 2 threads, then 2 **gmon.out** files will be generated, one per thread, using the naming convention as described earlier. You cannot enable thread level profiling by compiling an application with **-pg** in AIX 5.2 or earlier and running it in AIX 5.3. To enable thread level profiling you must compile that application with **-pg** in AIX 5.3.

The **gprof** command produces three items:

1. First, a flat profile is produced similar to that provided by the **prof** command. This listing gives total execution times and call counts for each of the functions in the program, sorted by decreasing time. The times are then propagated along the edges of the call graph. Cycles are discovered, and calls into a cycle are made to share the time of the cycle.
2. A second listing shows the functions sorted according to the time they represent, including the time of their call-graph descendents. Below each function entry are its (direct) call-graph children, with an indication of how their times are propagated to this function. A similar display above the function shows how the time of the function and the time of its descendents are propagated to its (direct) call-graph parents.
3. Cycles are also shown, with an entry for the cycle as a whole and a listing of the members of the cycle and their contributions to the time and call counts of the cycle.

Note: If the input to **gprof** contains thread level profiling data (AIX 5.3 format **gmon.out** file), then the **gprof** command produces the above 3 items for every thread, starting with a cumulative report, followed by per thread reported (sorted in the ascending order of thread IDs).

The **grpof** command can also be used to analyze the execution profile of a program on a remote machine. This can be done by running the **gprof** command with the **-c** option on the call graph profile file (**gmon.out** by default) to generate a file (**gprof.remote** by default) which can then be processed on a remote machine. If a call graph profile file other than **gmon.out** is to be used, the call graph profile file name(s) should be specified after **-c** *Filename* and the executable name. *Filename* must be specified if the **GPROF** environment variable's **file** attribute is set to **multi**; multiple gmon.out files are created, with one

gmon.out file for each PID when the executing program forks. The **-x** option can be used on the remote machine to process the **gprof.remote** (by default) file to generate profile reports.

Profiling with the fork and exec Subroutines

Profiling using the **gprof** command is problematic if your program runs the **fork** or **exec** subroutine on multiple, concurrent processes. Profiling is an attribute of the environment of each process, so if you are profiling a process that forks a new process, the child is also profiled. However, both processes write a **gmon.out** file in the directory from which you run the parent process, overwriting one of them. The **tprof** command is recommended for multiple-process profiling. In AIX 5.3, you can use **file:mutli** to avoid destroying the **gmon.out** file of the parent process, **file:multi** using the AIX 5.3 naming convention to generate the **gmon.out** files, hence the child processes **gmon.out** file will not have the same name as that of the parent, which will avoid overwrites.

For versions previous to AIX 5.3: If you must use the **gprof** command, one way around this problem is to call the **chdir** subroutine to change the current directory of the child process. Then, when the child process exits, its **gmon.out** file is written to the new directory. The following example demonstrates this method:

```
cd /u/test   # current directory containing forker.c program
pg forker.c
main()
{
int i, pid;
static char path[]="/u/test2";
pid=fork();             /* fork a child process */
if(pid==0) {            /* Ok, this is the child process */
   chdir (path);        /* create new home directory so
                           gmon.out isn't clobbered! */
   for (i=0; i<30000; i++) sub2();  /* 30000 calls to sub2
                                       in child profile */
   }
else                   /* Parent process... leave gmon.out
                          in current directory */
   for (i=0;i<1000; i++) sub1(pid);   /* 1000 calls to sub1
                                         in parent profile */
}
int sub1(pid)   /* silly little function #1, called
                   by parent 1000 times */
int pid;
{
int i;
printf("I'm the parent, child pid is %i.\n",pid);
}
int sub2()     /* silly little function #2, called
                  by child 30,000 times */
{
printf("I'm the child.\n");
}
cc -pg forker.c -o forker   # compile the program
mkdir /u/test2              # create a directory for childi
                              to write gmon.out in
forker >/dev/null           # Throw away forker's many,
                              useless output lines
gprof forker   >parent.out  # Parent process's gmon.out is
                              in current directory
gprof forker ../test2/gmon.out >child.out
                            # Child's gmon.out is in test2
                              directory
```

At this point, if you compare the two **gprof** command output listings in directory `test`, `parent.out`, and `child.out`, you see that the `sub1` subroutine is called 1,000 times in the parent and 0 times in the child, while the `sub2` subroutine is called 30,000 times in the child and 0 times in the parent.

Processes that run the **exec** subroutine do not inherit profiling. However, the program executed by the **exec** subroutine should be profiled if it was compiled with the **-pg** option. As with the preceding `forker.c` example, if both the parent and the program run by the **exec** subroutine program are profiled, one overwrites the other's **gmon.out** file unless you use the **chdir** subroutine in one of them.

Profiling without Source Code

If you do not have source for your program, you can profile using the **gprof** command without recompiling. You must, however, be able to relink your program modules with the appropriate compiler command (for example, **cc** for C). If you do not recompile, you do not get call frequency counts, although the flat profile is still useful without them. As an added benefit, your program runs almost as fast as it usually does. The following explains how to profile:

```
cc -c dhry.c         # Create dhry.o without call counting code.
cc -pg dhry.o -L/lib -L/usr/lib -o dhryfast
                     # Re-link (and avoid -pg libraries).
dhryfast             # Create gmon.out without call counts.
gprof >dhryfast.out  # You get an error message about no call counts
                     #  -- ignore it.
```

A result of running without call counts is that some quickly executing functions (which you know had to be called) do not appear in the listing at all. Although nonintuitive, this result is normal for the **gprof** command. The **gprof** command lists only functions that were either called at least once, or which registered at least one clock tick. Even though they ran, quickly executing functions often receive no clock ticks. Since call-counting was suspended, these small functions are not listed at all. (You can get call counts for the runtime routines by omitting the **-L** options on the **cc -pg** command line.)

Using Less Real Memory

Profiling with the **gprof** command can cause programs to page excessively since the **-pg** option dedicates pinned real-memory buffer space equal to one-half the size of your program's text. Excessive paging does not affect the data generated by profiling, since profiled programs do not generate ticks when waiting on I/O, only when using the CPU. If the time delay caused by excessive paging is unacceptable, we recommend using the **tprof** command.

Flags

-b	Suppresses the printing of a description of each field in the profile.
-c *Filename*	Creates a file that contains the information needed for remote processing of profiling information. Do not use the **-c** flag in combination with other flags.
-E *Name*	Suppresses the printing of the graph profile entry for routine *Name* and its descendants, similar to the **-e** flag, but excludes the time spent by routine *Name* and its descendants from the total and percentage time computations. (**-E** *MonitorCount* **-E** *MonitorCleanup* is the default.)
-e *Name*	Suppresses the printing of the graph profile entry for routine *Name* and all its descendants (unless they have other ancestors that are not suppressed). More than one **-e** flag can be given. Only one routine can be specified with each **-e** flag.
-F *Name*	Prints the graph profile entry of the routine *Name* and its descendants similar to the **-f** flag, but uses only the times of the printed routines in total time and percentage computations. More than one **-F** flag can be given. Only one routine can be specified with each **-F** flag. The **-F** flag overrides the **-E** flag.
-f *Name*	Prints the graph profile entry of the specified routine *Name* and its descendants. More than one **-f** flag can be given. Only one routine can be specified with each **-f** flag.
-g *Filename*	Writes call graph information to the specified output *filename*. It also suppresses the profile information unless the **-p** flag is used.
-i *Ffilename*	Writes the routine index table to the specified output *filename*. If this flag is not used, the index table goes either at the end of the standard output, or at the bottom of the filename(s) specified with the **-p** and **-g** flags.
-L *PathName*	Uses an alternate pathname for locating shared objects.
-p *Filename*	Writes flat profile information to the specified output filename. It also suppresses the call graph information unless the **-g** flag is used.

-s	Produces the **gmon.sum** profile file, which represents the sum of the profile information in all the specified profile files. This summary profile file may be given to subsequent executions of the **gprof** command (using the **-s** flag) to accumulate profile data across several runs of an **a.out** file.
-x *Filename*	Retrieves information from *Filename* (a file created with the **-c** option) to generate profile reports. If *Filename* is not specified, the **gprof** command searches for the default **gprof.remote** file.
-z	Displays routines that have zero usage (as indicated by call counts and accumulated time).

Examples

1. To obtain profiled output, enter:

   ```
   gprof
   ```

2. To get profiling output from a command run earlier and possibly moved, enter:

   ```
   gprof -L/home/score/lib runfile runfile.gmon
   ```

   This example uses the given **runfile.gmon** file for sample data and the **runfile** file for local symbols, and checks the **/u/score/lib** file for loadable objects.

3. To profile the sample program **dhry.c**:
   a. Recompile the application program with the **cc -pg** command, as follows:

      ```
      cc -pg dhry.c -o dhry # Re-compile to produce gprof output.
      ```

   b. Run the recompiled program. A file named `gmon.out` is created in the current working directory (not the directory in which the program executable resides).

      ```
      dhry    # Execute program to generate ./gmon.out file.
      ```

   c. Run the **gprof** command in the directory with the `gmon.out` file to produce the CALL-GRAPH and FLAT PROFILE reports.

      ```
      gprof >gprof.out     # Name the report whatever you like
      vi gprof.out         # Read flat profile first.
      ```

   d. To generated thread level profiling granularity, export the GPROF environment variable as follows, and run the application, do the following:

      ```
      export GPROF=profile:thread
      dhry   # Execute program to generate ./gmon.out file which has thread level granularity
      ```

   e. To generate per process **gmon.out** file with a prefix of `mygmon`, do the following:

      ```
      export GPROF=file:multi,filename:mygom
      dhry  # Execute program to generate ./gmon-dhry-2468.out
      ```

   f. To generate per thread **gmon.out** file, with a scaling factor of 10, with a file name prefixed as `tgmon`, do the following:

      ```
      export GPROF=profile:thread,file:multithread,scale:10,filename:tgmon
      dhry # Execute program to generate ./tgmon-dhry-2468-Pthread215.out
      ```

   g. To see only flat profile report from the `gmon-dhry-2468.out`, do the following:

      ```
      gprof -p fprofile.out ./dhry ./gmon-dhry-2468.out
      ```

   h. To see only call graph profile report from the `gmon-dhry-2468.out`, do the following:

      ```
      gprof -g callgraph.out ./dhry ./gmon-dhry-2468.out
      ```

4. To use the remote processing feature of **gprof** command:
   a. Recompile the application program with **cc -pg** command:

      ```
      cc -pg thread.c -o thread  -lpthread
      ```

   b. Enable thread level profiling granularity and use a different name for **gmon.out**:

      ```
      export GPROF=profile:thread,filename:mygmon
      ```

   c. Run the recompiled program. A file named **mygmon.out** is created in the current working directory (not the directory in which the program executable resides):

      ```
      thread    # Execute program to generate mygmon.out file.
      ```

d. Use the **-c** flag to generate the **my.remote** file, which can then be taken to a remote machine for processing:

```
gprof -c my.remote thread mygmon.out
```

e. On a remote machine, use the **-x** flag to extract information from the **my.remote** file:

```
gprof -x my.remote
```

Throughout this description of the **gprof** command, most of the examples use the C program **dhry.c**. However, the discussion and examples apply equally to FORTRAN, Pascal, or COBOL modules by substituting the appropriate compiler name in place of the C compiler, **cc**, and the word *subroutine* for the word *function*. For example, the following commands show how to profile a FORTRAN program named `matrix.f`:

```
xlf -pg matrix.f -o matrix # FORTRAN compile of matrix.f program
matrix                     # Execute with gprof profiling,
                           #   generating gmon.out file
gprof > matrix.out         # Generate profile reports in
                           #   matrix.out from gmon.out
vi matrix.out              # Read flat profile first.
```

Files

a.out	Name list and text space
gmon.out	Dynamic call graph and profile
gmon.sum	Summarized dynamic call graph and profile
gprof.remote	File for remote profiling
/usr/ucb/gprof	Contains the **gprof** command.
/usr/ccs/bin/gprof	Contains the **gprof** command

Related Information

The **cc** command, **prof** command.

The **exit** subroutine, **monitor** subroutine, **profil** subroutine.

Monitoring and tuning commands and subroutines in the *Performance management*.

The Commands in *Operating system and device management*.

The Subroutines Overview in *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

---

grap Command

Purpose

Typesets graphs to be processed by the **pic** command.

Syntax

grap [**-l**] [**-T** *Name*] [**—**] [*File* ...]

Description

The **grap** command processes grap language input files and generates input to the **pic** command. The grap language is a language for typesetting graphs. A typical command line is:

```
grap  File  |  pic  |  troff  |  Typesetter
```

Graphs are surrounded by the **.G1** and **.G2 troff** command requests. Data enclosed by these requests are scaled and plotted, with tick marks automatically supplied. Commands exist to modify the frame, add labels, override the default ticks, change the plotting style, define coordinate ranges and transformations, and include data from files. In addition, the **grap** command provides the same loops, conditionals, and macroprocessing as the **pic** command.

Grap language files contain grap programs. A grap program is written in the form:

```
.G1
grap Statement
grap Statement
grap Statement
.G2
```

Parameter

File Specifies grap language files (grap programs) to be processed by the **grap** command for input to the **pic** command.

grap Statements Summary

Following is a summary of the grap statements you can use to create a grap program:

frame Defines the frame that surrounds the graph. The syntax is:

```
frame [ht Expression] [wid Expression] [[Side] LineDescription]
```

The attributes are defined as follows:

- *Side*: `top`, `bot`, `left`, `right`
- *LineDescription*: `solid`, `invis`, `dotted [Expression]`, `dashed [Expression]`

Height defaults to 2 inches, width defaults to 3 inches, sides default to solid. If `side` is omitted, the *linedesc* applies to the entire frame.

label Places a label on a specified side of the graph. The syntax is:

```
label Side StringList ... Shift
```

The attributes are defined as follows:

- *Shift*: `left`, `right`, `up`, or `down` *expression*
- *StringList*: str ... `rjust`, `ljust`, `above`, `below [size (+)Expression] ...`
- *String*: `"..."`

coord Defines an overriding system. The syntax is:

```
coord [Name] [x Expression,Expression] [y  Expression,Expression] [[log x] [log y] [log log]]
```

ticks Places tick marks on one side of the frame. The syntax is:

```
ticks side [[in] [out] [Expression]] [Shift] [TickLocations]
```

The attributes are defined as follows:

- *Shift*: `left`, `right`, `up`, `down Expression`
- *TickLocations*: at [*Name*] *Expression* [*String*], *Expression* [*String*], ... from [*Name*] *Expression* to *Expression* [by [*Operation*] *Expression*] *String*

If no ticks are specified, they will be provided automatically; `ticks off` suppresses automatic ticks.

grid Produces grid lines along (that is, perpendicular to) the named side. The syntax is:

```
grid Side [LineDescription] [Shift] [TickLocations]
```

Grids are labeled by the same mechanism as ticks.

plot Places text at a point. The syntax is:

```
StartList at Point plot Expression [Start] at Point
```

The attributes are defined as follows:

- *StringList*: str ... `rjust`, `ljust`, `above`, `below` `[size +)Expression] ...`
- *Point*: `[Name] Expression Expression`

line Draws a line or arrow from one point to another. The syntax is:

```
{line | arrow} from Point to Point [LineDescription]
```

The attributes linedesc are defined as follows:

- *Point*: [*Name*] *Expression Expression*
- *LineDescription*: `solid`, `invis`, `dotted [Expression]`, `dashed Expression]`

circle Draws a circle. The syntax is:

```
circle at Point [radius Expression]
```

The radius is in inches; the default size is small.

draw Defines a sequence of lines. The syntax is:

```
draw [Name] at Point[LineDescription]
```

next Continues a sequence. The syntax is:

```
next [Name] at Point [LineDescription]
```

new Starts a new sequence. The syntax is:

```
new [Name] at Point [LineDescription]
```

numberlist Creates a line from a given set of numbers. The numbers are treated as points *x*, *y*1, *y*2, and so on; and plotted at the single *x* value. The syntax is:

```
number x, y1, y2 ...
```

for Creates a loop. The syntax is:

```
for Variable {from | =} Expression to Expression [by [arithmetic or multiplicative operator] Expression] do X Anything X
```

X is any single character that does not appear in the string. If `X` is a left brace {, then the string may contain internally balanced braces followed by a right brace}. The text `Anything` is repeated as the `Variable` takes on values from the first `Expression` to the second `Expression`.

if Creates a conditional evaluation. The syntax is:

```
if Expression then X Anything X [else X Anything X]
```

define Provides the same macroprocessor that Priority Interrupt Controller (PIC) does. The syntax is:

```
define MacroName X Anything X
```

copy Copies a file; includes the current contents of the file. The syntax is:

```
copy Filename
```

copy-thru Copies the file through the macro.

```
copy Filename thru MacroName
```

Each number or quoted string is treated as an argument. Copying continues until end of file or the next `.G2`. The optional clause `until String` causes copying to stop when a line whose first field is `String` occurs.

The following statement copies subsequent lines through the macro:

```
copy thru MacroName
```

In all cases, you can specify the macro by inline rather than by name:

```
copy thru x MacroBody x
```

sh Passes text through to the UNIX shell. The syntax is:

```
sh x Anything x
```

The variable `Anything` is scanned for macros. The **pid** macro is built-in. It is a string consisting of the process identification number; you can use it to generate unique file names.

pic Passes text through to **pic** with the `pic` removed. Variables and macros are not evaluated. Lines beginning with a period (that are not numbers) are passed through literally, under the assumption that they are **troff** commands.

graph Defines a new graph named *Picname*, and resets all coordinate systems. The syntax is:

```
graph Picname [pic-text]
```

If **graph** commands are used in a grap program, the statement after the `.G1` must be a **graph** command. You can use the `pic-text` to position this graph relative to previous graphs by referring to their Frames as in the following example.

```
graph First
 ...
graph Second with .Frames.w at First.Frame.e + [0.1,0]
```

Macros and expressions in `pic-text` are not evaluated. `Picnames` must begin with a capital letter according to **pic** syntax.

print Writes on `stderr` as grap processes its input. This statement can be helpful in debugging. The syntax is:

```
print [Expression | String]
```

grap Language Conventions

The following conventions apply:

- The `#` (pound sign) introduces a comment. The comment ends automatically at the end of a line.
- Statements that continue for more than one line must be preceded by a \ (backslash character) at the beginning of each new line.
- Multiple statements appearing on one line must be separated by semicolons.
- The grap language ignores blank lines.
- Predefined strings include `bullet`, `plus`, `box`, `star`, `dot`, `times`, `htick`, `vtick`, `square`, and `delta`.
- Built-in functions available in grap include `log` (base 10), `exp` (base 10), `int`, `sin`, `cos`, `atan2`, `sqrt`, `min`, `max`, and `rand`.

Flags

-l Stops the **grap** command from looking for the **/usr/lib/dwb/grap.defines** library file of macro definitions.

-T*Name* Specifies the value of the *Name* variable as the **grap** command output device. The default value is **-Tibm3816**.

- - (Double dash) Indicates the end of flags.

File

/usr/lib/dwb/grap.defines	Contains definitions of standard plotting characters.

Related Information

The **pic** command.

---

greek Command

Purpose

Converts English-language output from a Teletype Model 37 workstation to output for other workstations.

Syntax

greek [**-T** *Name*]

Description

The **greek** command reinterprets the Teletype Model 37 character set, including reverse and half-line motions, for display on other workstations. It simulates special characters, when possible, by overstriking. The **greek** command reads standard input and writes to standard output.

Flags

-T*Name*	Uses the specified workstation name. If you omit the **-T** flag, the **greek** command attempts to use the workstation specified in the **$TERM** environment variable. The value of the *Name* variable can be any one of the following:
300	DASI 300
300-12	DASI 300 in 12-pitch
300s	DASI 300s
300s-12	DASI 300s, in 12-pitch
450	DASI 450
450-12	DASI 450, in 12-pitch
2621	Hewlett-Packard 2621, 2640, and 2645
2640	Hewlett-Packard 2621, 2640, and 2645
2645	Hewlett-Packard 2621, 2640, and 2645
4014	Tektronix 4014
hp	Hewlett-Packard 2621, 2640, and 2645
tek	Tektronix 4014.

Environment Variables

$TERM	Specifies a workstation name.

Related Information

The **eqn** command, **hp** command, **mm** command, **neqn** command, **nroff** command,, **troff** command.

---

grep Command

Purpose

Searches for a pattern in a file.

Syntax

grep [**-E** | **-F**] [**-i**] [**-h**] [**-H**] [**-L**] [**-r** | **-R**] [**-s**] [**-v**] [**-w**] [**-x**] [**-y**] [[[**-b**] [**-n**]] | [**-c** | **-l** | **-q**]] [**-p** [S*eparator*]] { [**-e** *PatternList* ...] [**-f** *PatternFile* ...] | *PatternList* ... } [*File* ...]

Description

The **grep** command searches for the pattern specified by the *Pattern* parameter and writes each matching line to standard output. The patterns are limited regular expressions in the style of the **ed** or **egrep** command. The **grep** command uses a compact non-deterministic algorithm.

The **grep** command displays the name of the file containing the matched line if you specify more than one name in the *File* parameter. Characters with special meaning to the shell ($, *, [, |, ^, (,), \ ) must be in quotation marks when they appear in the *Pattern* parameter. When the *Pattern* parameter is not a simple string, you usually must enclose the entire pattern in single quotation marks. In an expression such as `[a-z]`, the - (minus sign) cml specifies a range, according to the current collating sequence. A collating sequence may define equivalence classes for use in character ranges. If no files are specified, **grep** assumes standard input.

Notes:

1. Do not run the **grep** command on a special file because it produces unpredictable results.
2. Input lines should not contain the NULL character.
3. Input files should end with the newline character.
4. The newline character will not be matched by the regular expressions.
5. Although some flags can be specified simultaneously, some flags override others. For example, the **-l** option takes precedence over all other flags. And if you specify both the **-E** and **-F** flags, the last one specified takes priority.

Flags

-b Precedes each line by the block number on which it was found. Use this flag to help find disk block numbers by context. The **-b** flag cannot be used with input from stdin or pipes.

-c Displays only a count of matching lines.

-E Treats each pattern specified as an extended regular expression (ERE). A NULL value for the ERE matches every line.

> **Note:** The **grep** command with the **-E** flag is the same as the **egrep** command, except that error and usage messages are different and the **-s** flag functions differently.

-e *PatternList* Specifies one or more search patterns. This works like a simple pattern but is useful when the pattern begins with a - (minus). Patterns should be separated by a new-line character. A NULL pattern can be specified by two adjacent new-line characters or a quotation mark followed by a new-line character (″\n). Each pattern is treated like a basic regular expression (BRE) unless the **-E** or **-F** flag is also specified. Multiple **-e** and **-f** flags are accepted by **grep**. All of the specified patterns are used when matching lines, but the order of evaluation is unspecified.

-F Treats each specified pattern as a string instead of a regular expression. A NULL string matches every line.

> **Note:** The **grep** command with the **-F** flag is the same as the **fgrep** command, except that error and usage messages are different and the **-s** flag functions differently.

-f *PatternFile* Specifies a file containing search patterns. Each pattern should be separated by a new-line character, and an empty line is considered a NULL pattern. Each pattern is treated like a basic regular expression (BRE), unless the **-E** or **-F** flag is also specified.

-h	Prevents the name of the file containing the matching line from being appended to that line. Suppresses file names when multiple files are specified.
-H	If the **-r** or **-R** option is specified and a symbolic link referencing a file of type directory is specified on the command line, **grep** will search the files of the directory referenced by the symbolic link and all the files in the file hierarchy below it.
-i	Ignores the case (uppercase or lowercase) of letters when making comparisons.
-l	Lists just the names of files (once) which contain matching lines. Each file name is separated by a new-line character. If standard input is searched, a path name of `(StandardInput)` is returned. The **-l** flag with any combination of the **-c** and **-n** flags behaves like the **-l** flag only.
-L	If the **-r** or **-R** option is specified and a symbolic link referencing a file of type directory is specified on the command line or encountered during the traversal of a file hierarchy, **grep** shall search the files of the directory referenced by the symbolic link and all the files in the file hierarchy below it. If both **-H** and **-L** are specified, the last option specified on the command line takes effect.
-n	Precedes each line with the relative line number in the file. Each file starts at line 1, and the line counter is reset for each file processed.
-p[*Separator*]	Displays the entire paragraph containing matched lines. Paragraphs are delimited by paragraph separators, as specified by the *Separator* parameter, which are patterns in the same form as the search pattern. Lines containing the paragraph separators are used only as separators; they are never included in the output. The default paragraph separator is a blank line.
-q	Suppresses all writing to standard output, regardless of matching lines. Exits with a zero status if an input line is selected. The **-q** flag with any combination of the **-c**, **-l** and **-n** flags behaves like the **-q** flag only.
-r	Searches directories recursively. By default, links to directories are followed.
-R	Searches directories recursively. By default, links to directories are not followed.
-s	Suppresses error messages ordinarily written for nonexistent or unreadable files. Other error messages are not suppressed.
-v	Displays all lines not matching the specified pattern.
-w	Does a word search.
-x	Displays lines that match the specified pattern exactly with no additional characters.
-y	Ignores the case of letters when making comparisons.
PatternList	Specifies one or more patterns to be used during the search. The patterns are treated as if they were specified using the **-e** flag.
File	Specifies a name of a file to be searched for patterns. If no *File* variable is given, the standard input is used.

Exit Status

This command returns the following exit values:

0	A match was found.
1	No match was found.
>1	A syntax error was found or a file was inaccessible (even if matches were found).

Examples

1. To use a pattern that contains some of the pattern-matching characters *, ^, ?, [,], \(, \), \{, and \}, enter:

   ```
   grep  "^[a-zA-Z]"  pgm.s
   ```

   This displays every line in `pgm.s` whose first character is a letter.
2. To display all lines that do not match a pattern, enter:

```
grep   -v  "^#" pgm.s
```

This displays every line in `pgm.s` whose first character is not a `#` (pound sign).

3. To display all lines in the `file1` file that match either the `abc` or `xyz` string, enter:

```
grep   -E  "abc|xyz"  file1
```

4. To search for a `$` (dollar sign) in the file named `test2`, enter:

```
grep \\$ test2
```

The `\\` (double backslash) characters are necessary in order to force the shell to pass a `\$` (single backslash, dollar sign) to the **grep** command. The `\` (single backslash) character tells the **grep** command to treat the following character (in this example the `$`) as a literal character rather than an expression character. Use the **fgrep** command to avoid the necessity of using escape characters such as the backslash.

5. To search recursively through **/tmp** to find files which have the word `IBM` without recursing through links pointing to directories, type:

```
grep –R IBM /tmp
```

OR

```
grep –r -H IBM /tmp
```

6. To search recursively through **/tmp** to find files which have the word `IBM` and recurse through links as well, type:

```
grep –r IBM /tmp
```

OR

```
grep -R -L IBM /tmp
```

Files

/usr/bin/grep	Contains the **grep** command.

Related Information

The **ed** command, **egrep** command, **fgrep** command, **sed** command.

Files in *Operating system and device management*.

Input and output redirection in *Operating system and device management*.

National Language Support in *AIX Version 6.1 National Language Support Guide and Reference*.

Shells in *Operating system and device management*.

---

groups Command

Purpose

Displays group membership.

Syntax

groups [*User...*]

Description

The **groups** command by default writes to standard ouput the group membership of the current process. If multiple users are specified as command parameters, then the group membership for each user will be displayed.

The **groups** command will continue its operation with the next user in the parameter list after issuing a warning message if the user given is not found in the user database.

Security

Access Control: This program should be installed as a normal user program in the Trusted Computing Base.

Examples

To display the group membership of users listed in the parameter list, enter:

```
$ groups sys root lp adm
sys : sys
root : system bin sys security cron audit lp
lp : lp printq
adm : adm
```

Files

/usr/bin/groups	Contains the **groups** command
/usr/ucb/groups	Symbolic link to the **groups** command
/etc/group	Group file; contains group IDs
/etc/ogroup	Previous version of the group file
/etc/passwd	Password file; contains user IDs
/etc/opasswd	Previous version of the password file.

Related Information

The **getty** command, **login** command, **setgroups** command, **su** command, **tsm** command.

---

grpck Command

Purpose

Verifies the correctness of a group definition. This document describes both the AIX **grpck** command and the System V **grpck** command.

Syntax

grpck { **-n** | **-p** | **-t** | **-y** } { **ALL** | *Group ...* }

Description

The **grpck** command verifies the correctness of the group definitions in the user database files by checking the definitions for **ALL** the groups or for the groups specified by the *Group* parameter. If more than one group is specified, there must be a space between the groups.

> **Note:** This command writes its messages to **stderr**.

You must select a flag to indicate whether the system should try to fix erroneous attributes. The following attributes are checked:

name Checks the uniqueness and composition of the group name. The group name must be a unique string of eight bytes or less. It cannot begin with a + (plus sign), a : (colon), a - (minus sign), or a ~ (tilde). It cannot contain a colon (:) in the string and cannot be the **ALL** or **default** keywords. No system fix is possible.

groupID Checks the uniqueness and composition of the group ID. The ID must not be null and must consist of decimal digits only. No system fix is possible.

users Checks the existence of the users listed in the group database files. If you indicate that the system should fix errors, it will delete all the users that are not found in the user database files.

adms Checks the existence of the users listed as group administrators in the group database files. If you indicate that the system should fix errors, it will delete all the administrators that are not found in the user database files.

admin Checks for a valid admin attribute for each group in the **/etc/security/group** file. No system fix is available.

Generally, the **sysck** command calls the **grpck** command as part of the verification of a trusted-system installation. In addition, the root user or a member of the security group can enter the command.

The **grpck** command checks to see if the database management security files (**/etc/passwd.nm.idx**, **/etc/passwd.id.idx**, **/etc/security/passwd.idx**, and **/etc/security/lastlog.idx**) files are up-to-date or newer than the corresponding system security files. Please note, it is alright for the **/etc/security/lastlog.idx** to be not newer than **/etc/security/lastlog**. If the database management security files are out-of-date, a warning message appears indicating that the root user should run the **mkpasswd** command.

Flags

-n Reports errors but does not fix them.
-p Fixes errors but does not report them.
-t Reports errors and asks if they should be fixed.
-y Fixes errors and reports them.

Security

Access Control: This command should grant execute (x) access to the root user and members of the security group. The command should be **setuid** to the root user and have the **trusted computing base** attribute.

Files Accessed:

Mode	File
r	/etc/passwd
r	/etc/security/user
rw	/etc/security/group
rw	/etc/group

Auditing Events:

Event	Information
GROUP_User	user, groups, attribute | error, status
GROUP_Adms	user, groups, attribute | error, status

Attention RBAC users and Trusted AIX users: This command can perform privileged operations. Only privileged users can run privileged operations. For more information about authorizations and privileges,

see Privileged Command Database in *Security*. For a list of privileges and the authorizations associated with this command, see the **/etc/security/privcmds** database file.

Examples

1. To verify that all the group members and administrators exist in the user database, and have any errors reported (but not fixed), enter:

   ```
   grpck  -n ALL
   ```

2. To verify that all the group members and administrators exist in the user database and to have errors fixed, but not reported, enter:

   ```
   grpck  -p ALL
   ```

3. To verify the uniqueness of the group name and group ID defined for the `install` group, enter:

   ```
   grpck  -n install
   ```

   OR

   ```
   grpck  -t install
   ```

   OR

   ```
   grpck  -y install
   ```

   The **grpck** command does not correct the group names and IDs. Therefore, the **-n, -t** and **-y** flags report problems with group names and group IDs, but do not correct them.

Files

/usr/sbin/grpck	Contains the **grpck** command.
/etc/passwd	Contains the basic attributes of users.
/etc/security/user	Contains the extended attributes of users.
/etc/group	Contains the basic attributes of groups.
/etc/security/group	Contains the extended attributes of groups.

Related Information

The **pwdck** command, **sysck** command, **usrck** command.

For more information about the identification and authentication of users, discretionary access control, the trusted computing base, and auditing, refer to *Security*.

System V grpck command

Syntax

/usr/sysv/bin/grpck

Description

The **/usr/sysv/bin/grpck** command verifies the correctness of the group definitions in the user database files by checking the definitions for ALL the groups.This **/usr/sysv/bin/grpck** command is a System V version of the existing **grpck** command in **/usr/sbin/**.This command calls the **/usr/sbin/grpck** command with **-n** and **ALL** options.

Exit Status

0 Successful completion.

>0 An error occurred.

Examples

1. To verify that all the group members and administrators exist in the user database, and have any errors reported (but not fixed), enter:

   ```
   /usr/sysv/bin/grpck
   ```

Files

/usr/sysv/bin/grpck
Contains the System V version of the **grpck** command.

Related Information

The **/usr/sbin/grpck** command.

---

grpsvcsctrl Command

Purpose

Starts the group services subsystems.

Syntax

grpsvcsctrl { **–a** | **–s** | **–k** | **–d** | **–c** | **–u** | **–t** | **–o** | **–h** }

Description

The **grpsvcsctrl** command starts the group services subsystems. This 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 **grpsvcs**. A daemon is associated with each subsystem. From an operational point of view, the group services subsystem group is organized as follows:

Subsystem	group services
Subsystem group	**grpsvcs**
SRC subsystem	**grpsvcs** — associated with the **hagsd** daemon. The subsystem name on the nodes is **grpsvcs**. The **grpsvcs** subsystem on each node is associated with the cluster to which the node belongs.
Daemon	**hagsd** — provides the majority of the group services functions.

The **grpsvcsctrl** script is not normally run from the command line. It is normally called by the startup command during installation of the cluster.

The **grpsvcsctrl** script provides a variety of controls for operating the group services subsystems:

- Adding, starting, stopping, deleting, and cleaning up the subsystems
- Turning tracing on and off

Before performing any of these functions, the script obtains the current cluster name.

Adding the subsystem: 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 the **grpsvcs** subsystem is stopped.
2. It gets the port number for the **grpsvcs** subsystem for this cluster from the global object data manager (ODM) and makes sure the port number is set in the **/etc/services** file. The range of valid port numbers is 10000 to 10100, inclusive.
3. The service name that is entered in the **/etc/services** file is **grpsvcs.***cluster_name*.
4. It removes the **grpsvcs** subsystem from the SRC (in case it is still there).
5. It adds the **grpsvcs** subsystem to the SRC. The cluster name is configured as a daemon parameter on the **mkssys** command.

Starting the subsystem: When the **-s** flag is specified, the control script uses the **startsrc** command to start the group services subsystem, **grpsvcs**.

Stopping the subsystem: When the **-k** flag is specified, the control script uses the **stopsrc** command to stop the group services subsystem, **grpsvcs**.

Deleting the subsystem: When the **-d** flag is specified, the control script uses the **rmssys** command to remove the group services subsystem from the SRC. The control script operates as follows:

1. It makes sure the **grpsvcs** subsystem is stopped.
2. It removes the **grpsvcs** subsystem from the SRC using the **rmssys** command.
3. It removes the port number from the **/etc/services 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 grpsvcs** command.
2. 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.

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.

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. The default is 5000 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.

Flags

-a Adds the subsystem.

-s Starts the subsystems.

-k Stops the subsystems.

-d Deletes the subsystems.

-c Cleans the subsystems (that is, deletes them from all system partitions).

-u Removes the group services subsystem from all partitions.

-t Turns tracing on for the subsystems.

-o Turns tracing off for the subsystems.

-h Writes the script's usage statement to standard output.

Security

You must be running with an effective user ID of **root**.

Exit Status

0 Indicates the successful completion of the command.

1 Indicates that an error occurred.

Restrictions

This script is valid in an HACMP environment only.

Standard Output

When the **-h** flag is specified, this command's usage statement is written to standard output.

Standard Error

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

Examples

1. To add the group services subsystems to the SRC, enter:
   ```
   grpsvcsctrl -a
   ```
2. To start the group services subsystems, enter:
   ```
   grpsvcsctrl -s
   ```
3. To stop the group services subsystems, enter:
   ```
   grpsvcsctrl -k
   ```
4. To delete the group services subsystems from the SRC, enter:
   ```
   grpsvcsctrl -d
   ```
5. To clean up the group services subsystems, enter:
   ```
   grpsvcsctrl -c
   ```
6. To turn tracing on for the group services daemon **hagsd**, enter:
   ```
   grpsvcsctrl -t
   ```
7. To turn tracing off for the group services daemon **hagsd**, enter:
   ```
   grpsvcsctrl -o
   ```

Location

/usr/sbin/rsct/bin/grpsvcsctrl Contains the **grpsvcsctrl** script

Files

/var/ha/log/grpsvcs_nodenum_instnum.cluster_name
Contains the log of the **hagsd** daemons on the nodes

The file name includes these variables:

nodenum
is the node number on which the daemon is running

instnum
is the instance number of the daemon

cluster_name
is the name of the cluster in which the daemon is running

Related Information

Commands: **lssrc**, **mkssys**, **rmssys**, **startsrc**, **stopsrc**

Daemons: **hagsd**

---

gssd Daemon

Purpose

Services kernel requests for GSS operations.

Syntax

/usr/sbin/gssd

Description

Some NFS security methods, such as Kerberos 5, are provided under a more general mechanism called General Security Services, or GSS. In AIX, GSS services are provided by a library in the IBM Network Authentication Service (NAS) fileset. NAS is shipped on the expansion pack. The **gssd** daemon makes these GSS services available to the NFS server kernel code. If the **gssd** daemon is not running, then efforts to access files via NFS using GSS security methods such as Kerberos 5 will fail. The **gssd** daemon registers using RPC program number 400234.

The **gssd** daemon is started and stopped with the following System Resource Controller (SRC) commands:

```
startsrc -s gssd
stopsrc -s gssd
```

Files

/etc/nfs/hostkey Specifies **keytab** file location and host principal in the following format:

```
path to keytab file
host principal
```

/etc/nfs/princmap Specifies mappings to host principals in the following format:

```
principal1 alias1 alias2 alias3
principal2 alias1
```

The aliases can be IP addresses or hostnames; the principal must match the host key maintained by kerberos.

---

ha_star Command

Purpose

Processes high availability event.

Syntax

ha_star [**-C**]

Description

The **ha_star** command is the generic high availability handling command. It is automatically invoked by the operating system through **/etc/rc.ha_star** when a CPU predictive failure is reported by the firmware.

If **ha_star** is invoked without flags, only new events are handled. If **ha_star** does not find any new event, it exits.

When running, **ha_star** handles all new events, even those which arrive while **ha_star** is handling already existing events. Only one instance of **ha_star** can be running at any given time. Should a second instance of **ha_star** be launched, it exits.

The operating system invokes **ha_star** when a high availability event is reported. The event handling may fail or it may be cancelled (for example, by signals). Aborted or cancelled events are held in memory within the kernel. When the cause of the abort has been corrected, then the event handling can be retried. This is when **ha_star** is invoked manually by the system administrator.

The **ha_star** command generates error or failure error log entries.

Description by Event Type

The **ha_star** command is invoked by the operating system to deallocate a CPU when a predictive processor failure event is detected. This deallocation may fail because some threads remain bound to the CPU being deallocated. In some cases, system administrators can fix the condition which led to the failure of the deallocation. For example, they may be able to identify and stop applications with threads bound to the last logical CPU.

The **-C** flag indicates that the high availability event to be resumed is a CPU deallocation event.

Flags

-C	Specifies that the event to be restarted is a CPU deallocation.

Files

/usr/sbin/ha_star	Contains the **ha_star** command.

Related Information

The Dynamic Processor Deallocation in the *AIX Version 6.1 General Programming Concepts: Writing and Debugging Programs*.

The Enabling Dynamic Processor Deallocation in the *Operating system and device management*.

---

ha.vsd Command

Purpose

Queries and controls the activity of the rvsd daemon of the recoverable virtual shared disk subsystem.

Syntax

ha.vsd {**adapter_recovery [on** | **off]** | **debug [off]** | **mksrc** | **query** | **quorum** *n* | **qsrc** | **refresh [noquorum]** | **reset** | **reset_quorum** | **rmsrc** | **start** | **stop** | **trace [off]**}

Description

Use this command to display information about the recoverable virtual shared disk subsystem, to change the number of nodes needed for quorum, and to change the status of the subsystem.

Flags

–a Specifies all virtual shared disks.

–v *vsd_name_list*
Specifies one or more virtual shared disk names, separated by commas.

–n *node_list* Specifies one or more node numbers, separated by commas.

Parameters

adapter_recovery [on | off]
Enables or disables communication adapter recovery. The default is **on**.

The recoverable virtual shared disk subsystem must be restarted for this operand to take effect.

debug [off] Specify **debug** to redirect the recoverable virtual shared disk subsystem's standard output and standard error to the console and cause the recoverable virtual shared disk subsystem to not respawn if it exits with an error. (You can use the **lscons** command to determine the current console.)

The recoverable virtual shared disk subsystem must be restarted for this operand to take effect.

Once debugging is turned on and the recoverable virtual shared disk subsystem has been restarted, **ha.vsd trace** should be issued to turn on tracing.

Use this operand under the direction of your IBM service representative.

Note: The default when the node is booted is to have standard output and standard error routed to the console. If debugging is turned off standard output and standard error will be routed to **/dev/null** and all further trace messages will be lost. You can determine if debug has been turned on by issuing **ha.vsd qsrc**. If debug has been turned on the return value will be:

```
action = "2"
```

mksrc Uses **mkssys** to create the recoverable virtual shared disk subsystem.

query Displays the current status of the recoverable virtual shared disk subsystem in detail.

quorum *n* Sets the value of the quorum, which is the total number of nodes that must join the group before the virtual shared disks will be activated. Usually, quorum is defined as a majority of the nodes that are defined as virtual shared disk nodes in an RSCT peer domain, but this command allows you to override that definition.

The Recoverable virtual shared disk subsystem must be in the active state when you issue this command. This is not a persistent change.

qsrc Displays the System Resource Controller (SRC) configuration of the Recoverable virtual shared disk daemon.

refresh [noquorum]
Uses the **refresh** command to asynchronously start a refresh protocol to all running recoverable virtual shared disk subsystems. The quorum will be reset before the refresh occurs, unless **noquorum** is specified. Use **ha.vsd query** to check for completion. The following items are refreshed in the device driver:
1. Nodes that have been added or deleted

2. Virtual shared disks that have been added or deleted
3. Changed attribute size_in_MB for virtual shared disks

reset Stops and restarts the recoverable virtual shared disk subsystem.

reset_quorum
Resets the default quorum.

rmsrc Uses **rmssys** to remove the recoverable virtual shared disk subsystem.

start Starts the recoverable virtual shared disk subsystem.

stop Stops the recoverable virtual shared disk subsystem.

trace [off] Requests or stops tracing of the recoverable virtual shared disk subsystem. The recoverable virtual shared disk subsystem must be in the active state when this command is issued.

This operand is only meaningful after the **debug** operand has been used to send standard output and standard error to the console and the recoverable virtual shared disk subsystem has been restarted.

Security

You must have **root** authority to run this command.

Exit Status

0 Indicates the successful completion of the command.

nonzero Indicates that an error occurred.

Restrictions

You must issue this command from a node that is online in the peer domain. To bring a peer domain online, use the **startrpdomain** command. To bring a particular node online in an existing peer domain, use the **startrpnode** command. For more information on creating and administering an RSCT peer domain, refer to *RSCT Administration Guide* .

Examples

1. To stop the recoverable virtual shared disk 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 the RSCT peer domain, enter:

```
ha.vsd quorum 5
```

   The system returns the message:

```
Quorum has been changed from 8 to 5.
```

3. To query the rvsd subsystem, enter:

```
ha.vsd query
```

   The system displays a message similar to the following:

```
Subsystem         Group            PID     Status
      rvsd              rvsd             18320   active
      rvsd(vsd): quorum= 9/4, active=1, state=idle, isolation=member,
```

```
NoNodes=10, lastProtocol=nodes_failing,
adapter_recovery=on, adapter_status=up,
RefreshProtocol has never been issued from this node,
Running function level 4.1.0.0.
```

where:

quorum Is the number of total nodes or server nodes that must join the group before virtual shared disks will be activated. In the system output above, `quorum 9/4` indicates the total number of nodes (9) and the number of server nodes (4).

active Indicates the activation status of the group that is being joined:

- **0:** the group is not active (quorum has not been met).
- **1:** the group is active and the shared disks have been activated.

state Indicates the current protocol that is running.

isolation Indicates the group membership status

- **isolated:** a group ″join″ has not been proposed.
- **proposed:** a group ″join″ has been proposed.
- **member:** we are a member (provider) of the group.

NoNodes Indicates the number of nodes that have joined the group

lastProtocol Indicates the last protocol that was run across the group.

adapter_recovery
Indicates communication adapter recovery support:

- **on:** adapter recovery is enabled.
- **off:** adapter recovery is disabled.

adapter_status
Indicates communication adapter status:

- **up:** the adapter is up.
- **down:** the adapter is down.
- **unknown:** the adapter status is unknown.

RefreshProtocol ...
Indicates whether a refresh protocol has been issued from this node. If so, the date and time of success or error will be displayed.

Running function level
Indicates the function level that the subsystem is running, in version, release, modification, fix level format (vrmf). (Coexistence with lower levels of the subsystem, may restrict us to running at a reduced function level.)

Location

/opt/rsct/vsd/bin/ha.vsd

Related Information

Commands: **ha_vsd**

ha_vsd Command

Purpose

Starts and restarts the Recoverable virtual shared disk subsystem. This includes configuring virtual shared disks and activating the recoverability subsystem.

Syntax

ha_vsd [**reset**]

Description

Use this command to start the recoverable virtual shared disk software after you install it, or, with the **reset** option, to stop and restart the program.

Flags

–a Specifies all virtual shared disks.

–v *vsd_name_list*
Specifies one or more virtual shared disk names, separated by commas.

–n *node_list* Specifies one or more node numbers, separated by commas.

Parameters

reset Stops and restarts the recoverable virtual shared disk subsystem.

Security

You must have **root** authority to run this command.

Exit Status

0 Indicates the successful completion of the command.

1 Indicates that an error occurred.

Restrictions

You must issue this command from a node that is online in the peer domain. To bring a peer domain online, use the **startrpdomain** command. To bring a particular node online in an existing peer domain, use the **startrpnode** command. For more information on creating and administering an RSCT peer domain, refer to *RSCT Administration Guide* .

Examples

To stop the recoverable virtual shared disk subsystem and restart it, enter:

```
ha_vsd reset
```

Location

/opt/rsct/vsd/bin/ha_vsd

Related Information

Commands: **ha.vsd**

haemd Daemon

Purpose
Observes resource variable instances that are updated by resource monitors and generates and reports events to client programs.

Syntax
haemd

Description
The **haemd** (event manager) 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 every node of a cluster. 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 **emsvcsctrl** command. If you must start or stop the daemon directly, use the **emsvcsctrl** command.

When SRC creates the **haemd** daemon, the actual program started is **haemd_HACMP**. The **haemd_HACMP** program, after collecting information needed by the daemon, then runs the **haemd** program. In other words, the **haemd_HACMP** program is replaced by the **haemd** program in the process created by SRC.

For more information about the event manager daemon, see the **emsvcsctrl** command.

Implementation Specifics
This daemon is part of Reliable Scalable Cluster Technology (RSCT) fileset for AIX.

Location
/usr/sbin/rsct/bin/haemd Location of the **haemd** daemon

Related Information
Commands: **emsvcsctrl**, **haemd_HACMP**

haemd_HACMP Command

Purpose
Startup program for the event manager daemon.

Syntax
haemd_HACMP [**-d** *trace_arg*]

Description
The **haemd_HACMP** command is the startup program for the **haemd** daemon. When the event management subsystem is configured in the system resource controller (SRC) by the **emsvcsctrl** command, **haemd_HACMP** is specified as the program to be started.

This program can only be invoked by the SRC. To start the event management subsystem, use the **emsvcsctrl** command.

Flags

-d *trace_arg*

Should only be used under the direction of the IBM Support Center. The possible trace arguments are the same as for the **haemtrcon** command, except for **reg** and **dinsts**. To use this flag, the **emsvcs** subsystem definition in the SRC must be changed using the **chssys** command with the **-a** flag. The daemon must then be stopped and restarted.

Restrictions

This command is valid in an HACMP environment only.

Implementation Specifics

This script is part of the Reliable Scalable Cluster Technology (RSCT) fileset.

Location

/usr/sbin/rsct/bin/haemd_HACMP

Location of the **haemd_HACMP** program

Related Information

Commands: **emsvcsctrl**, **haemd**, **haemtrcon**

---

haemqvar Command

Purpose

Queries resource variables.

Syntax

haemqvar [**-H** *domain* | **-S** *domain*] [**-c** | **-d** | **-i**] [**-f** *file*] [**-h**] [*class var rsrcID* [*″*]]

Description

The haemqvar command queries the Event Management subsystem for information about resource variables. By default, the command writes to standard output the definitions for all resource variables in the current SP™ domain, that is, the current SP system partition as defined by the SP_NAME environment variable. If SP_NAME is not set the default system partition is used. The -S flag can be used to specify another SP domain (system partition). To query variables in an HACMP domain, use the -H flag. For an SP domain, the domain flag argument is a system partition name. For an HACMP domain, the domain flag argument is an HACMP cluster name. When the -H flag is specified, the command must be executed on one of the nodes in the HACMP/ES cluster.

The following information is reported for each resource variable definition:

- Variable Name
- Value Type
- Data Type
- SBS Format (if data type is Structured Byte String)
- Initial Value
- Class
- Locator

- Variable Description
- Resource ID and its description
- Default Expression (if defined) and its description

Because the default behavior of this command can produce a large amount of output, standard output should be redirected to a file.

If the -d flag is specified only the resource variable name and a short description are written to standard output, one name and description per line.

If the -c flag is specified the current values of all resource variables instances are written to standard output, one per line. The line of output contains the location of the resource variable instance (node number), the resource variable name, the resource ID of the instance and the resource variable instance value. If the resource variable is a Structured Byte String (SBS) data type, then the value of each SBS field is reported.

The -i flag reports the same information as the -c flag except that the value of the variable instance is the last known value rather than the current value. The -i flag is useful for determining what resource variable instances exist.

For both the -c and the -i flags, if an error is encountered in obtaining information about a resource variable instance, the output line contains an error message, symbolic error codes, the location of where the error originated (if it can be determined), the resource variable name and the resource ID.

To return information about specific resource variables, specify the class, var and rsrcID operands. These operands can be repeated to specify additional resource variables. In addition, the var and rsrcID operands can be wildcarded to match a number of resource variables. Note that null string operands or an asterisk must be quoted in the shells.

If class is not a null string, then all variables in the specified class, as further limited by the var and rsrcID arguments, are targets of the query. If class is a null string, then variables of all classes, as further limited by the var and rsrcID arguments, are targets of the query. The var argument can be wildcarded in one of two ways:

1. Specify the variable name as a null string
2. Truncate the name after any component

When the resource variable name is wildcarded in the first manner, then all resource variables, as further limited by the class and rsrcID arguments, are targets of the query. When the resource variable name is wildcarded in the second manner, all resource variables whose high-order (leftmost) components match the var argument, as further limited by the class and rsrcID arguments, are targets of the query.

All resource variable instances, or definitions if neither the -c nor the -i flags are specified, of the variables specified by the class and var arguments that match the rsrcID argument are the targets of the query.

If neither the -c nor the -i flags are specified, the rsrcID argument is a semicolon-separated list of resource ID element names. If either the -c or the -i flags is specified, the rsrcID argument is a semicolon-separated list of name/value pairs. A name/value pair consists of a resource ID element name followed by an equal sign followed by a value of the resource ID element. An element value may consist of a single value, a range of values, a comma-separated list of single values or a comma-separated list of ranges. A range takes the form a-b and is valid only for resource ID elements of type integer (the type information can be obtained from the variable definition). There can be no blanks in the resource ID.

A resource ID element is wildcarded by specifying its value as the asterisk character. Only variables that are defined to contain the elements, and only the elements, specified in the rsrcID argument are targets of the query. If any element of the resource ID consists of the asterisk character, rather than a name/value

pair (or just a name if querying for definitions), all variables that are defined to contain at least the remaining specified elements are targets of the query. The entire resource ID is wildcarded if it consists of only the asterisk character; all instances of all resource variables, as further limited by the class and var arguments, are targets of the query.

Note that the rsrcID argument must be quoted in the shells if it contains semicolons or asterisks.

The class, var and rsrcID operands can be placed in a file, one set of operands per line, instead of being specified as command arguments. Use the -f flag to specify the name of the file to the command. If the -f flag is used, any operands to the command are ignored. Within the file, null strings are specified as two adjacent double quotation marks. A completely wildcarded resource ID can either be a single asterisk (*) or an asterisk in double quotation marks (″*″). The arguments must be separated by blank spaces or tabs on each line.

Some examples of using wildcards in the rsrcID argument follow. For these examples, assume the class and var arguments are null strings. If either the class or var arguments or both are not null strings, targets for the query are restricted accordingly. In the first three examples, all variables whose resource IDs are defined to contain the elements NodeNum, VG and LV, and only those elements, are matched.

1. In this example, only one instance is matched:
   ```
   NodeNum=5;VG=rootvg;LV=hd4
   ```
2. In this example, one instance from each node is matched:
   ```
   NodeNum=*;VG=rootvg;LV=hd4
   ```
3. In this example, all instances of the matching resource variables are matched:
   ```
   NodeNum=*;VG=*;LV=*
   ```
4. In this example, all variables whose resource IDs are defined to contain only the element NodeNum are matched. The instances matched are associated with node 9:
   ```
   NodeNum=9
   ```
5. In this example, the same set of variables are matched, but all instances of each variable are matched:
   ```
   NodeNum=*
   ```
6. In this example, all variables whose resource IDs are defined to contain elements NodeNum and VG, as well as zero or more additional elements, are matched. The instances matched are associated with node 9:
   ```
   NodeNum=9;VG=*;*
   ```
7. In this example, all variables whose resource IDs are defined to contain the element NodeNum, as well as zero or more additional elements, are matched. All instances of the variables are matched:
   ```
   NodeNum=*;*
   ```

Given the flexibility in specifying resource variables for query, it is possible that no resource variable instance or resource variable definition will match. If there is no match appropriate error information is reported, either in the form described above or as follows.

If the specification of the class, var or rsrcID arguments are in error, the output line contains an error message, symbolic error codes and the specified class name, resource variable name, and resource ID.

Flags

-H *domain*
: Queries resource variables in the HACMP domain specified by *domain*.

-S *domain*
: Queries resource variables in the SP domain specified by *domain*.

-c
: Queries current resource variable values.

-d
: Queries resource variable definitions but produces short form output.

-i Queries instances of resource variables.

-f *file* Queries resource variables specified in *file*.

-h Displays a usage statement.

Parameters

class Specifies the name of the resource variable class or a null string.

var Specifies the name of the resource variable or a null string.

rsrcID Specifies a resource ID or an asterisk.

Security

You must have root privilege and write access to the SDR to run this command.

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.

Exit Status

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.

Restrictions

This command is valid in a PSSP environment only.

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.

Examples

1. To obtain the definitions of all resource variables in the current cluster and place the output in a file, enter:

   ```
   haemqvar -H HAcluster > vardefs.out
   ```

2. To obtain a short form list of all resource variables whose resource IDs contain the element VG, in the HACMP cluster named HAcluster, enter:

   ```
   haemqvar -H HAcluster -d "" "" "VG;*"
   ```

3. To obtain resource variables whose resource IDs contain only the elements VG and NodeNum, enter:

   ```
   haemqvar -H HAcluster -d "" "" "VG;NodeNum"
   ```

Location

/usr/sbin/rsct/bin/haemqvar Location of the **haemqvar** command

Files

/usr/sbin/rsct/install/config/haemloadlist
Contains the default configuration data for the Event Management subsystem

Related Information

Commands: **haemcfg**, **SDRCreateObjects**, **SDRDeleteObjects**

Files: **haemloadlist**

For information about the System Data Repository (SDR) classes and attributes for the Event Management configuration database, see *RSCT Event Management Programming Guide and Reference*

---

haemtrcoff Command

Purpose

Turns tracing off for the Event Manager daemon.

Syntax

haemtrcoff -s *subsys_name* **-a** *trace_list*

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.

Flags

-s *subsys_name*
Specifies the name of the Event Management subsystem. On a node this is emsvcs. 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.

Parameters

The following trace arguments can 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 expression evaluation.

rdi Stops tracing the reliable daemon (internal) interface.

sched Stops tracing the internal scheduler.

shm Stops tracing shared memory management activity.

all Stops tracing all activities.

all_but_msgs
Stops tracing all activities except for messages. Message activity is defined by the msgs argument.

Security

You must have root privilege and write access to the SDR to run this command.

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.

Exit Status

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.

Restrictions

Do not use this command during normal operation. 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.

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.

Examples

1. To turn off all tracing for the Event Management subsystem on one of the cluster nodes, log in to the node and enter:

   ```
   haemtrcoff -s emsvcs -a all
   ```

2. To turn off all tracing of initialization and configuration for the Event Management subsystem on a cluster node, log in to the node and enter:

   ```
   haemtrcoff -s emsvcs -a init,config
   ```

Location

/usr/sbin/rsct/bin/haemtrcoff Location of the **haemtrcoff** command

Files

/var/ha/log/em.trace.cluster_name
Contains the trace log of the **haemd** daemon on the cluster named *cluster_name*

/var/ha/log/em.msgtrace.cluster_name
Contains message trace output from the Event Manager daemon on the cluster named *cluster_name*

Related Information

Commands: **haemtrcon**

Daemons: **haemd**

Scripts: **emsvcsctrl**

---

haemtrcon Command

Purpose

Turns tracing on for the event manager daemon.

Syntax

haemtrcon -s *subsys_name* **-a** *trace_list*

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. When used, the **regs**, **dinsts**, **iolists**, and **olists** parameters perform a one-time trace. The specified information is placed in the trace log, but no further tracing is done.

Flags

-s *cluster_name*
Specifies the name of the event management subsystem. On a node, *cluster_name* is **emsvcs**. This flag and parameter must be specified.

-a *trace_list*
Specifies a list of trace parameters. Each parameter specifies the type of activity for which tracing is to be turned on. At least one parameter must be specified. If more than one parameter is specified, the parameters must be separated by commas. The list may not include blanks.

Parameters

The following trace parameters can 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 expression evaluation.

rdi Traces the reliable daemon (internal) interface.

sched Traces the internal scheduler.

shm Traces shared memory management activity.

all Traces all activities.

all_but_msgs
Stops tracing all activities except for messages. Message activity is defined by the msgs argument.

regs Traces currently registered events.

dinsts Traces all resource variable instances known to the daemon.

iolists Traces immediate observation lists

olists Traces observation lists

Restrictions

Do not use this command during normal operation. 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.

Implementation Specifics

This command is part of the Reliable Scalable Cluster Technology (RSCT) fileset.

Examples

1. To turn on all tracing for the event management subsystem on one of the cluster nodes, log in to the node and enter:
   ```
   haemtrcon -s emsvcs -a all
   ```
2. To turn on all tracing of initialization and configuration for the event management subsystem on a cluster node, log in to the node and enter:
   ```
   haemtrcon -s emsvcs -a init,config
   ```

Location

/usr/sbin/rsct/bin/haemtrcon Location of the **haemtrcon** command

Related Information

Commands: **haemtrcoff**

Daemons: **haemd**

Scripts: **emsvcsctrl**

haemunlkrm Command

Purpose

Unlocks and starts a resource monitor.

Syntax

haemunlkrm -s *subsys_name* **-a** *resmon_name*

Description

If the event management daemon cannot successfully start a resource monitor after three attempts within a two-hour interval, or if the daemon has successfully connected to the instances of a resource monitor *n* times within a two-hour interval, the resource monitor is "locked" and no further attempts are made to start it or to connect to any of its instances. *n* is **3** in an HACMP/ES cluster. Once the cause of the failure is determined and the problem corrected, the **haemunlkrm** command can be used to unlock the resource monitor and attempt to start it or connect to the resource monitor instances.

The status of the event manager daemon, as displayed by the **lssrc** command, indicates whether a resource monitor is locked.

Flags

-s *subsys_name*
: Specifies the name of the event management subsystem. On a node, *subsys_name* is **emsvcs**. This flag and parameter must be specified.

-a *resmon_name*
: Specifies the name of the resource monitor to unlock and start.

Parameters

The following trace parameters can 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 expression evaluation.

rdi Traces the reliable daemon (internal) interface.

sched Traces the internal scheduler.

shm Traces shared memory management activity.

all Traces all activities.

all_but_msgs
Stops tracing all activities except for messages. Message activity is defined by the msgs argument.

regs Traces currently registered events.

dinsts Traces all resource variable instances known to the daemon.

iolists Traces immediate observation lists

olists Traces observation lists

Security

You must have root privilege and write access to the SDR to run this command.

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.

Exit Status

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.

Restrictions

Do not use this command during normal operation. 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.

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.

Examples

1. This example applies to unlocking a resource monitor on a node.
   If the output of the **lssrc** command indicates that the program resource monitor **IBM.PSSP.harmpd** is locked, correct the condition that prevented the resource monitor from being started and enter:
   ```
   haemunlkrm -s emsvcs -a IBM.PSSP.harmpd
   ```

Location

/usr/sbin/rsct/bin/haemunlkrm
Location of the **haemunlkrm** command

Files

/var/ha/log/em.trace.*cluster_name*
Contains the trace log of the haemd daemon on the cluster named *cluster_name*.

/var/ha/log/em.msgtrace.cluster_name
Contains message trace output from the event manager daemon on the cluster named *cluster_name*.

Related Information

Commands: **haemtrcoff**

Daemons: **haemd**

Scripts: **emsvcsctrl**

---

hagsd Daemon

Purpose

Observes resource variable instances that are updated by resource monitors and generates and reports events to client programs.

Syntax

hagsd [**-a**] [**-s**] [**-k**] [**-d**] [**-c**] [**-u**] [**-t**] [**-o**] [**-r**] [**-h**] *daemon_name*

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 the nodes of a cluster. This daemon provides most of the services of the subsystem. *daemon_name* specifies the name used by the daemon to name log files and identify its messages in the AIX error log.

One instance of the **hagsd** daemon executes on each cluster node. The **hagsd** daemon is under the control of the system resource controller (SRC).

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 **grpsvcsctrl** command, which is in turn called by the cluster startup process. If you must start or stop the daemon directly, use the **startsrc** or **stopsrc** command.

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.

–u Unconfigures the subsystems 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.

Parameters

daemon_name
Specifies the name used by the daemon to name log files and identify its messages in the AIX error log.

Security

You must have **root** privilege to run this script.

Exit Status

0 Indicates the successful completion of the command.

1 Indicates that an error occurred.

Restrictions

This command is valid in a PSSP environment only.

Standard Output

When the **-h** flag is specified, this command's usage statement is written to standard output.

Standard Error

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

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 unconfigure the group servicess subsystem from all system partitions, on the control workstation, enter:
   ```
   hagsctrl -u
   ```
7. 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:
   ```
   hagsctrl -t
   ```
8. 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:
   ```
   hagsctrl -o
   ```

Location

/usr/sbin/rsct/bin/hagsd Contains the **hagsd** daemon

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.

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.

Related Information

Commands: **grpsvcsctrl**

---

hagsns Command

Purpose

Gets group services name server information.

Syntax

hagsns [**–h** *host*] [**–c**] **–g** *group_name*

hagsns [**–h** *host*] [**–c**] **–s** *subsystem_name*

hagsns [**–h** *host*] [**–c**] **–p** *subsystem_pid*

Description

Use the **hagsns** command to query the status of the group services nameserver.

Flags

–c Forces the output as ″English_only.″ If the **–c** flag is not specified, the daemon's locale will be used for the output.

–g *group_name*
Specifies a group of subsystems to get status for. The command is unsuccessful if the *group_name* variable is not contained in the subsystem object class.

–h *host*
Specifies the host to obtain name server status for.

–p *subsystem_pid*
Specifies a particular instance of the *subsystem_pid* to obtain name server status for.

–s *subsystem_name*
Specifies a subsystem to get status for. The *subsystem_name* variable can be the actual subsystem name or the synonym name for the subsystem. The command is unsuccessful if the *subsystem_name* variable is not contained in the subsystem object class.

Parameters

daemon_name
Specifies the name used by the daemon to name log files and identify its messages in the AIX error log.

Security

You must have **root** authority to run this command.

Exit Status

0 Indicates that the command completed successfully.

a non-zero value
Indicates that an error occurred.

Restrictions

This command is valid in a PSSP environment only.

Standard Output

When the **-h** flag is specified, this command's usage statement is written to standard output.

Standard Error

This command writes error messages, as necessary, to standard error.

Examples

To get domain information from the group services subsystem, enter:

```
hagsns -c -s cthags
```

or

```
hagsns -s cthags
```

The output will look like this:

```
HA GS NameServer Status
NodeID=1.16, pid=14460, domainID=6.14, NS established,CodeLevel=GSLevel(DRL=8)
NS state=kCertain, protocolInProgress=kNoProtocol,outstandingBroadcast=KNoBcast
Process started on Jun 19 18:34:20, (10d 20:19:22) ago, HB connection took (19:14:9).
Initial NS certainty on Jun 20 13:48:45, (10d 1:4:57) ago, taking (0:0:15).
Our current epoch of Jun 23 13:05:19 started on (7d 1:48:23), ago.
Number of UP nodes: 12
List of UP nodes: 0 1 5 6 7 8 9 11 17 19 23 26
```

In this example, **domainID=6.14** means that node 6 is the name server (NS) node. The domain ID consists of a node number and an incarnation number. The incarnation number is an integer, incremented whenever the group services daemon is started. **NS established** means that the name server was established.

Location

/usr/sbin/rsct/bin/hagsns Contains the **hagsns** command

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.

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.

Related Information

Commands: **hagsvote**, **lssrc**, **nlssrc**

---

hagsvote Command

Purpose

Gets vote information for group services groups.

Syntax

hagsvote [**–h** *host*] [**–l**] [**–a** *argument*] [**–c**] **–g** *group_name*

hagsvote [**–h** *host*] [**–l**] [**–a** *argument*] [**–c**] **–s** *subsystem_name*

hagsvote [**–h** *host*] [**–l**] [**–a** *argument*] [**–c**] **–p** *subsystem_pid*

Description

Use the **hagsvote** command to query the status of voting protocols for group services.

Flags

–a Specifies a group services group name. This group name is different from that of the **–g** flag. In this case, the group was created from the client's first call to join the protocol.

–c Requests the canonical output of the group services voting information. The output is displayed in English regardless of the installed language locale. If **–c** is not specified, the daemon's locale will be used for the output.

–g *group_name*
Specifies a group of subsystems to get status for. The command is unsuccessful if the *group_name* variable is not contained in the subsystem object class.

–h *host*
Specifies the host name which is getting status.

–l Requests detailed output in "long" form.

–p *subsystem_pid*
Specifies a particular instance of the *subsystem_pid* variable to get the vote for.

–s *subsystem_name*
Specifies a subsystem to vote on. The *subsystem_name* variable can be the actual subsystem name or the synonym name for the subsystem. The command is unsuccessful if the *subsystem_name* variable is not contained in the subsystem object class.

Parameters

daemon_name
Specifies the name used by the daemon to name log files and identify its messages in the AIX error log.

Security

You must have **root** privilege to run this command.

Exit Status

0 Indicates the successful completion of the command.

non-zero
Indicates that an error occurred.

Restrictions

This command is valid in a PSSP environment only.

Standard Output

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

Standard Error

This command writes error messages, as necessary, to standard error.

Examples

1. To see information about the status of the voting protocol for the group **theSourceGroup** in long form, enter:

   ```
   hagsvote -ls cthags -a theSourceGroup (locale-dependent)
   ```

   The output will look like this:

   ```
   Number of groups: 4
   Group name [theSourceGroup] GL node [26] voting data:
   GL in phase [1] of n-phase protocol of type [Join].
   Local voting data:
   Number of providers: 1
   Number of providers not yet voted: 1 (vote not submitted).
   Given vote: [No vote value] Default vote: [No vote value]
   ProviderID Voted? Failed? Conditional?
   [101/26] No No Yes
   Global voting data:
   Number providers not yet voted: 1
   Given vote: [No vote value] Default vote: [No vote value]
   Nodes that have voted: []
   Nodes that have not voted: [26]
   ```

   The first line of the output means that the total number of groups is 4. The second line provides the group name and the group leader node (in this case 26). The remaining lines give the voting data:

   - The group leader is in phase 1 of a n-phase protocol.
   - The protocol is the Join protocol.
   - For the local node, it has 1 provider, the number of providers which have not voted yet is 1.
   - No default vote value is given and no vote value is given.
   - Under the line ″ProviderID Voted? Failed? Conditional?,″ ″[101/16] No No Yes,″ means that the provider ID is 101/26, not voted yet, not failed, but wait for the vote (so it is conditional).

   The output then shows the global voting status:

   - The number of providers that have not voted yet is 1.

- No vote value given yet, no default vote value.
- The nodes that have voted is none.
- The nodes that have not voted is node 26.

2. In the following example, the meaning of each line of output is the same as in the first example except that node 26 is the group leader node.

   ```
   hagsvote -ls cthags -a theSourceGroup -c (canonical form)
   ```

   The output will look like this:

   ```
   Number of groups: 4
   Group Name: theSourceGroup
   GL Node: 26 (I am GL)
   Current phase number of an n-phase protocol: 1
   Protocol name: [Join]
   Local voting data:
   Number of local providers: 1
   Number of local providers not yet voted: 1 (vote not submitted)
   Given vote: [No vote value] Default vote: [No vote value]Global voting data:
   Number of nodes in group: 1
   Number of global providers not yet voted: 1
   Given vote: [No vote value] Default vote: [No vote value]
   Nodes that have voted: []
   Nodes that have not voted: [26]
   ```

Location

/usr/sbin/rsct/bin/hagsvote Contains the **hagsvote** command

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.

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.

Related Information

Commands: **hagsns**, **lssrc**, **nlssrc**

---

halt or fasthalt Command

Purpose

Stops the processor.

Syntax

{ **halt** | **fasthalt** } [**-l**] [**-n**] [**-p**] [**-q**] [**-y**]

Description

The **halt** command writes data to the disk and then stops the processor. The machine does not restart. Only a root user should run this command. Do not use this command if other users are logged into the

system. If no other users are logged in, the **halt** command can be used. Use the **halt** command if you are not going to restart the machine immediately. When the message `....Halt completed....` is displayed, you can turn the power Off.

The **halt** command logs the shutdown using the **syslogd** command and places a record of the shutdown in **/var/adm/wtmp**, the login accounting file. The system also writes an entry into the error log which states that the system was shut down.

The **fasthalt** command stops the system by calling the **halt** command. The **fasthalt** command provides BSD compatibility.

Flags

-l Does not log the halt in the accounting file. The **-l** flag does not suppress accounting file update. The **-n** and **-q** flags imply the **-l** flag.

-n Prevents the **sync** before stopping.

-p Halts the system without a power down.

> **Note:** The **-p** flag will have no effect if used in combination with flags not requiring a permanent halt. Power will still be turned off if other operands request a delayed poweron and reboot

-q Causes a quick halt.

> **Note:** Running **halt** command with **-q** flag does not issue **sync**, so the system will halt immediately.

-y Halts the system from a dial-up operation.

Security

Attention RBAC users and Trusted AIX users: This command can perform privileged operations. Only privileged users can run privileged operations. For more information about authorizations and privileges, see Privileged Command Database in *Security*. For a list of privileges and the authorizations associated with this command, see the **/etc/security/privcmds** database file.

Examples

1. To halt the system without logging the halt in the accounting file, enter:
   ```
   halt -l
   ```
2. To halt the system quickly, enter:
   ```
   halt -q
   ```
3. To halt the system from a dial-up, enter:
   ```
   halt -y
   ```

Files

/etc/rc	Specifies the system startup script.
/var/adm/wtmp	Specifies the login accounting file.

Related Information

The **fastboot** command, **fsck** command, **rc** command, **shutdown** command, **sync** command.

The **syslogd** daemon.

hangman Command

Purpose

Starts the hangman word-guessing game.

Syntax

hangman [*File*]

Description

The **hangman** command chooses a word of at least seven letters from a standard dictionary. The *File* parameter specifies an alternate dictionary. You guess the word by guessing letters one at a time. You are allowed seven mistakes.

When you start hangman, the game displays:

```
guesses: word: ....... errors: 0/7
guess:
```

The `guesses` displays the letters you have used as guesses. Every letter you guess is listed after `guesses`. The `word: .......` displays the number of letters in the mystery word. In this case there are seven `.` (periods) so there are seven letters in the word. As you correctly guess letters, the game replaces the appropriate `.` with the correct letter. The `errors: 0/7` displays the number of incorrect guesses. You enter your letter guess at the `guess:` prompt. For example:

```
guesses: word: .......... errors: 0/7
guess: q
guesses: q word: .......... errors: 1/7
guess: a
guesses: aq word: .a....a... errors: 1/7
guess: b
guesses: abq word: .a....a... errors 2/7
guess: j
guesses: abjq word: .a....a... errors: 3/7
guess: s
guesses: abjqs word: .a....a..s errors: 3/7
guess: z
guesses: abjqsz word: .a....a..s errors: 4/7
guess: y
guesses: abjqsyz word: .a....a..s errors: 5/7
guess: k
guesses: abjkqsyz word: .a....a..s errors: 6/7
guess: x
the answer was calculates, you blew it
```

To quit the game, press the Interrupt (Ctrl-C) or End Of File (Ctrl-D) key sequence.

Files

/usr/games	Location of the system's games.

Related Information

The **arithmetic** command, **back** command, **bj** command, **craps** command, **fish** command, **fortune** command, **moo** command, **number** command, **quiz** command, **ttt** command, **turnoff** command, **turnon** command, **wump** command.

hatsoptions Command

Purpose

Controls topology services options on a node or a control workstation.

Syntax

hatsoptions [**-s**] [**-d**]

Description

Before this command can be executed, environment variable HB_SERVER_SOCKET must be set to the location of the UNIX-domain socket used by the topology services subsystem. The statement below can be used:

```
export HB_SERVER_SOCKET=/var/ha/soc/hats/server_socket.partition name
```

Alternatively, variable HA_SYSPAR_NAME can be set to the partition name.

The topology services daemon must be running in order for this command to be successful.

hatsoptions can be used to control a number of options in topology services. Option **-s** instructs the topology services daemon to reject messages that are apparently delayed. This can be used in very large system configurations, where messages are sometimes delayed in the network or in the sender and receiver nodes. Use this option only if the Time-Of-Day clocks are synchronized across all the nodes and the control workstation. Otherwise messages may be incorrectly discarded when the sender's Time-Of-Day clock is behind the receiver's.

Option **-d** instructs the topology services daemon not to reject messages that are apparently delayed. This is the default.

Flags

–s Instructs the topology services daemon to reject messages that are apparently delayed.

–d Instructs the topology services daemon not to reject messages that are apparently delayed (this is the default).

Security

You must have **root** privilege to run this command.

Exit Status

0 Indicates the successful completion of the command.

1 Indicates the command was unsuccessful.

Environment Variables

HB_SERVER_SOCKET

This environment variable should be set before this command can be executed. It must be set to the location of the UNIX-domain socket used by topology services clients to connect to the topology services daemon. This environment variable must be set to **/var/ha/soc/hats/server_socket.***partition name*.

HA_SYSPAR_NAME

If HB_SERVER_SOCKET is not set, then HA_SYSPAR_NAME must be set to the partition name.

Restrictions

This command is valid in a peer domain only.

Standard Output

When the **-h** flag is specified, this command's usage statement is written to standard output. All verbose messages are written to standard output.

Standard Error

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

Examples

To instruct the topology services daemon on the local node to start discarding apparently delayed messages, enter:

```
export HA_SYSPAR_NAME=partition1

/usr/sbin/rsct/bin/hatsoptions -s
```

Location

/usr/sbin/rsct/bin/hatsoptions
Contains the **hatsoptions** command

Files

/var/ha/soc/hats/server_socket.*partition name*

Related Information

Commands: **hatsctrl**, **hats**, **lssrc**, **startsrc**, **stopsrc**, **syspar_ctrl**

---

hash Command

Purpose

Remembers or reports command path names.

Syntax

To Add the Path of a Command to the Path Name List:

hash [*Command* ...]

To Clear Path Name List:

hash -r

Description

The **hash** command affects the way the current shell remembers a command's path name, either by adding a path name to a list or purging the contents of the list.

When no parameter or flag is specified, the **hash** command reports to standard output the contents of the path name list. The report includes the path name of commands in the current shell environment that were found by previous **hash** command invocations. The display may also contain those commands invoked and found through the normal command search process.

Note: Shell built-in commands are not reported by the **hash** command.

You can use the **-r** flag to clear the contents of the command path name list. Path names can also be cleared from the list by resetting the value of the **PATH** environment variable. In the simplest form, this would be achieved by entering:

```
PATH="$PATH"
```

If the *Command* parameter is used, the **hash** command searches for the path name of the specified command and adds this path to the list. Do not use a / (slash) when you specify the command.

Since the **hash** command affects the current shell environment, it is provided as a Korn shell or POSIX shell regular built-in command. If the **hash** command is called in a separate command execution environment, as in the following examples, it will not affect the command search process of the caller's environment:

```
nohup hash -r
find . -type f | xargs hash
```

Using the **hash** command is equivalent to using the **alias -t** command.

Flag

-r Clears the contents of the path name list.

Parameter

Command Specifies the *Command* to add to the path name list.

Exit Status

The following exit values are returned:

0 Successful completion.
>0 An error occurred.

Examples

1. To find the path name of the **wc** command and add it to the path name list, enter:

   ```
   hash wc
   ```

2. To clear the contents of the path name list, enter:

   ```
   hash -r
   ```

Files

/usr/bin/ksh Contains the Korn shell **hash** built-in command.
/usr/bin/hash Contains the **hash** command.

Related Information

The **alias** command, **bsh** command, **ksh** command.

---

head Command

Purpose

Displays the first few lines of a file.

Syntax

head [**-***Count* | **-c** *Number* | **-n** *Number*] [*File* ...]

Description

The **head** command writes to standard output a specified number of lines or bytes of each of the specified files, or of the standard input. If no flag is specified with the **head** command, the first 10 lines are displayed by default. The *File* parameter specifies the names of the input files. An input file must be a text file. When more than one file is specified, the start of each file will look like the following:

```
==> filename <==
```

To display a set of short files, identifying each one, enter:

```
example% head -9999 filename1 filename2...
```

Flags

-*Count*	Specifies the number of lines from the beginning of each specified file to be displayed. The *Count* variable must be a positive decimal integer. This flag is equivalent to the **-n** *Number* flag, but should not be used if portability is a consideration.
-c *Number*	Specifies the number of bytes to display. The *Number* variable must be a positive decimal integer.
-n *Number*	Specifies the number of lines from the beginning of each specified file to be displayed. The *number* variable must be a positive decimal integer. This flag is equivalent to the **-***Count* flag.

Exit Status

This command returns the following exit values:

0	Successful completion.
>0	An error occurred.

Examples

To display the first five lines of the `Test` file, enter:

```
head -5 Test
```

OR

```
head -n 5 Test
```

Related Information

The **tail** command.

Files in *Operating system and device management*.

Input and output redirection in *Operating system and device management*.

---

help Command

Purpose

Provides information for new users.

Syntax

help

Description

The **help** command presents a one-page display of information for new users. Information is available for the following topics:

- Concatenating or displaying files.
- Editing lines interactively.
- Sending and receiving mail.
- Reading system messages.
- Changing password file information.
- Identifying current users of the system.
- Sending messages to the other users on the system.
- Displaying the contents of directories.
- Viewing information on the Source Code Control System.
- Setting terminal modes.

Examples

To obtain help, type `help` at the command line.

Related Information

The **cat** command, **ex** command, **finger** command, **ls** command, **mail** command, **passwd** command, **sccshel**p command, **tset** command, **who** command, **write** command.

---

host Command

Purpose

Resolves a host name into an Internet address or an Internet address into a host name.

Syntax

host [**-n** [**-a**] [**-c** *Class*] [**-d**] [**-r**] [**-t** *Type*] [**-v**] [**-w**] [**-z**]] *Hostname* | *Address* [*Server*]

hostnew [**-a**] [**-c** *Class*] [**-d**] [**-r**] [**-t** *Type*] [**-v**] [**-w**] [**-z**] *Hostname* | *Address* [*Server*]

Description

The **/usr/bin/host** command returns the Internet address of a host machine when the *HostName* parameter is specified and the name of the host when the *Address* parameter is specified. Depending on the configuration of name resolution service, the **host** command may also display any aliases associated with the *HostName* parameter. Examples of name resolution services include **local**, **nis**, and **bind**.

If the local host is using the Domain Name Protocol, the local or remote name server database is queried before searching the local **/etc/hosts** file.

The **host** command may also return other name records found in the DNS (Domain Name System). The **-z** flag specifies this mode. The other flags allow for the customization of the query.

Flags

-a Equivalent to using ″**-v -t ***″

-c *Class* Specifies the class to look in when searching non-Internet data. Valid classes are:

IN Internet class

CHAOS
Chaos class

HESIOD
MIT Althena Hesiod class

ANY Wildcard (any of the above)

***** Wildcard (any of the above)

-d Turns on debugging mode.

-n Equivalent to issuing the **/usr/bin/hostnew** command. The **hostnew** command is the 5.2 version of the **host** command.

-r Disables recursive processing.

-t *Type* Specifies the type of record to query for. Valid types are:

A Host's Internet address

CNAME
Canonical name for an alias

HINFO Host CPU and operating system type

KEY Security Key Record

MINFO Mailbox or mail list information

MX Mail exchanger

NS Nameserver for the named zone

PTR Host name if the query is an Internet address; otherwise, the pointer to other information

SIG Signature Record

SOA Domain's ″start-of-authority″ information

TXT Text information

UINFO User information

WKS Supported well-known services

-v Verbose mode.

-w Waits forever for a reply from the DNS server.

-z Uses the new output that displays resource record information.

Parameters

Address Specifies the Internet address of the host machine to use in resolving the host name. The *Address* parameter must be a valid Internet address in dotted decimal format.

HostName Specifies the name of the host machine to use in resolving the Internet address. The *HostName* parameter can be either a unique host name or a well-known host name (such as *nameserver*, *printserver*, or *timeserver*, if these exist).

Server Specifies the nameserver to query.

Examples

1. To display the address of a host machine named `mephisto`, enter:

   ```
   host mephisto
   ```

   Information similar to the following is displayed:

   ```
   mephisto is 192.100.13.5, Aliases: engr, sarah
   ```

2. To display the host whose address is 192.100.13.1, enter:

```
host 192.100.13.1
```

   Information similar to the following is displayed:

```
mercutio is 192.100.13.1
```

3. To display the MX records for the domain named test.ibm.com, enter:

```
host -n -t mx test.ibm.com
```

   or

```
hostnew -t mx test.ibm.com
```

   Information similar to the following is displayed:

```
test.ibm.com mail is handled (pri=10) by test1.tt.ibm.com
test.ibm.com mail is handled (pri=10) by test2.aix.ibm.com
```

Files

/etc/hosts Contains the Internet Protocol (IP) name and addresses of hosts on the local network.

Related Information

The **hostname** command.

The **named** daemon.

Communications and networks in *Networks and communication management*.

---

hostent Command

Purpose

Directly manipulates address-mapping entries in the system configuration database.

Syntax

To Add an Address-to-Host Name Mapping

hostent -a *IPAddress* **-h** *"HostName..."*

To Delete an Address-to-Host Name Mapping

hostent -d *IPAddress*

To Delete All Address-to-Host Name Mappings

hostent -X

To Change an Address-to-Host Name Mapping

hostent -c *IPAddress* **-h** *"HostName..."* [**-i** *NewIPAddress*]

To Show an Address or Host Name in Colon Format

hostent -s { *IPAddress* | *"HostName"* } [**-Z**]

To Show all Address-to-Host Name Mappings in Colon Format

hostent -S [**-Z**]

Description

The **hostent** low-level command adds, deletes, or changes address-mapping entries in the system configuration database. Entries in the database are used to map an Internet Protocol (IP) address (local or remote) to its equivalent host names.

The **hostent** command can show one or all address-to-host name mapping entries in the **/etc/hosts** file. An Internet Protocol (IP) address of a given local or remote host may be associated with one or more host names. Represent an IP address in dotted decimal format. Represent a host name as a string with a maximum length of 255 characters, and use no blank characters. Each entry must be contained on one line. Multiple *HostNames* (or aliases) can be specified.

Note: Valid host names or alias host names must contain at least one alphabetic character. If you choose to specify a host name or alias that begins with an `x` followed by any hexadecimal digit (`0-f`), the host name or alias must also contain at least one additional letter that cannot be expressed as a hexadecimal digit. The system interprets a leading `x` followed by a hexadecimal digit as the base 16 representation of an address unless there is at least one character in the host name or alias that is not a hexadecimal digit. Thus, `xdeer` would be a valid host name, whereas `xdee` would not.

You can use the System application in Web-based System Manager (wsm) to change system characteristics. You could also use the System Management Interface Tool (SMIT) **smit hostent** fast path to run this command.

Flags

Note: The **-a**, **-d**, **-c**, and **-s** flags cannot be used together.

-a *IPAddress*	Adds an IP address-to-host name mapping entry for the given Internet Protocol address in the database. Specify the host names with the **-h** flag.
-c *IPAddress*	Changes an IP address-to-host name mapping entry in the database that corresponds to the given address specified by the *IPAddress* variable. Specify the changed host names with the **-h** flag. If you want to change the current IP address to a new address (*IPAddress*), use the **-i** flag.
-d *IPAddress*	Deletes the IP address-to-host name mapping entry in the database that corresponds to the given address specified by the *IPAddress* variable.
-h*"HostName..."*	Specifies a list of host names. Entries in the list should be separated by blanks. The **-h***"HostName..."* flag should be used with the **-a** flag. The **-c** flag may also require the **-h***"HostName..."* flag.
-i *NewIPAddress*	Specifies a new IP address. This flag is required by the **-c** flag if an existing IP address is to be replaced by the *NewIPAddress* variable.
-S	Shows all entries in the database.
-s*"HostName"*	Shows an IP address-to-host name mapping entry matching the host name specified by the *"HostName"* variable.
-s *IPAddress*	Shows an IP address-to-host name mapping entry matching the entry specified by the *IPAddress* variable.
-X	Deletes all IP address-to-host name mapping entries in the database.
-Z	Generates the output of the query in colon format. This flag is used when the **hostent** command is invoked from the SMIT usability interface.

Note: The **hostent** command does recognize the following addresses: .08, .008, .09, and .009. Addresses with leading zeros are interpreted as octal, and numerals in octal cannot contain 8s or 9s.

Security

Attention RBAC users and Trusted AIX users: This command can perform privileged operations. Only privileged users can run privileged operations. For more information about authorizations and privileges,

see Privileged Command Database in *Security*. For a list of privileges and the authorizations associated with this command, see the **/etc/security/privcmds** database file.

Examples

1. To add an entry in the database associating an address with a series of host names, enter the command in the following format:

   ```
   hostent  -a 192.100.201.7  -h "alpha bravo charlie"
   ```

   In example 1, the IP address `192.100.201.7` is specified as the address of the host that has a primary host name of `alpha` with synonyms of `bravo` and `charlie`.

   Note: If you attempt to use .08, .008, .09, or .009 in an address to add, you will get an error message that states `"IP Address Address already exists,"` although the address is not in the **/etc/hosts** file.
2. To show an entry in the database matching a host name, enter the command in the following format:

   ```
   hostent  -s alpha
   ```

   In example 2, the entry to be shown matches the host name `alpha`.
3. To change the IP address of an entry to a new IP address, enter the command in the following format:

   ```
   hostent  -c 192.100.201.7  -i 192.100.201.8
   ```

   In example 3, the old IP address is `192.100.201.7` and the new address is `192.100.201.8`.

Files

/etc/hosts	Contains host names and addresses for the network.

Related Information

The **hostname** command.

TCP/IP name resolution in *Networks and communication management*.

For information on installing the Web-based System Manager, see Chapter 2: Installation and System Requirements in *AIX Version 6.1 Web-based System Manager Administration Guide*.

---

hostid Command

Purpose

Sets or displays the identifier of the current local host.

Syntax

/usr/sbin/hostid [*HexNumber* | *InternetAddress* | *HostName*]

Description

The **/usr/sbin/hostid** command displays the identifier (either a unique host name or a numeric argument) of the current local host as a hexadecimal number. This numeric value is expected to be unique across all hosts and is commonly set to the address of the host specified by the *InternetAddress* or *HostName*

parameter. The root user can set the **hostid** command by specifying a hexadecimal number for the *HexNumber*, *InternetAddress*, or *HostName* parameter. The host identifier is set to the hostname by the **/etc/rc.net** file.

Parameters

HexNumber	Specifies a unique hexadecimal number representing the current local host.
InternetAddress	Specifies an Internet address representing the current local host.
HostName	Specifies a symbolic name that maps to a unique host.

Examples

1. To set the identifier of the local host to the local Internet address with the **hostid** command, enter the command in the following format:

   ```
   hostid   192.9.200.3
   0xc009c803
   ```

   The **hostid** command converts the Internet address `192.9.200.3` into the hexadecimal representation `0xc009c803`, and then sets the local host (your workstation connected to a network) to this address.
2. To display the identifier of the local host, enter:

   ```
   hostid
   0xc009c803
   ```

   The **hostid** command displays the identifier of the host as a hexadecimal number.

Related Information

The **hostname** command.

The **gethostid** subroutine, **sethostid** subroutine.

The **rc.net** file format.

TCP/IP addressing in *Networks and communication management*.

---

hostmibd Daemon

Purpose

Starts the **hostmibd** dpi2 sub-agent daemon as a background process.

Syntax

hostmibd [**-f** *File*] [**-d** [*Level*]] [**-h** *Hostname*] [**-c** *Community*]

Description

The **hostmibd** command starts the **hostmibd** dpi2 sub-agent. This command may only be issued by a user with root privileges or by a member of the system group.

The **hostmibd** daemon complies with the standard Simple Network Management Protocol Distributed Protocl Interface Version 2.0 defined by RFC 1592. It is acting as a dpi2 sub-agent to communicate with the dpi2 agent through dpiPortForTCP.0 (1.3.6.1.4.1.2.2.1.1.1.0) which is defined in RFC1592 section 3.1.

The Management Information Base (MIB) is defined by RFC 1155. The specific MIB variables **hostmibd** is managing are defined by RFC 2790. The actual MIB variables managed by **hostmibd** are the following six subtrees:

- hrSystem (1.3.6.1.2.1.25.1)
- hrStorage (1.3.6.1.2.1.25.2)
- hrDevice (1.3.6.1.2.1.25.3)
- hrSWRun (1.3.6.1.2.1.25.4)
- hrSWRunPerf (1.3.6.1.2.1.25.5)
- hrSWInstalled (1.3.6.1.2.1.25.6)

The **hostmibd** daemon is normally executed during system startup when **/etc/rc.tcpip** shell script is called.

The **hostmibd** daemon should be controlled using the System Resource Controller(SRC). Entering **hostmibd** at the command line is not recommended.

Use the following SRC commands to manipulate the **hostmibd** daemon:

startsrc
Starts a subsystem, group of subsystems, or a subserver.

stopsrc
Stops a subsystem, group of subsystems, or a subserver.

refresh
Causes a subsystem or group of subsystems to reread the appropriate configuration file.

lssrc Gets the status of a subsystem, group of subsystems, or a subserver. If the user issuing the long status form of the **lssrc** command is not the root user, no community name information is displayed.

Flags

-c *Community* Use specified community name. If **-c** flag is not specified, the default community name is 'public'.

-d *Level* Specifies tracing/debug level. The levels are:

- 0 = Least level
- 8 = DPI level 1
- 16 = DPI level 2
- 32 = Internal level 1
- 64 = Internal level 2
- 128 = Internal level 3

Add the numbers for multiple trace levels. The default level is 56 if the **-d** flag is specified but *Level* is not specified. If the **-d** flag is not specified, the default level is 0.

-f *File* Specifies a non-default configuration file. If the **-f** flag is not specified, the default configuration file is **/etc/hostmibd.conf**. See **/etc/hostmibd.conf** file for information on this file format.

-h *Host* Send request to specified host. If **-h** flag is not specified, the default destination host is 'loopback' (127.0.0.1).

Examples

1. To start the **hostmibd** daemon, enter a command similar to the following:

   ```
   startsrc -s hostmibd -a "-f /tmp/hostmibd.conf"
   ```

   This command starts the **hostmibd** daemon and reads the configuration file from **/tmp/hostmibd.conf**.
2. To stop the **hostmibd** daemon, normally enter:

   ```
   stopsrc -s hostmibd
   ```

This command stops the **hostmibd** daemon. The **-s** flag specified the subsystem that follows to be stopped.

3. To get the short status from the **hostmbid**, enter:

   ```
   lssrc -s hostmbid
   ```

   This command returns the name of the daemon, the process ID of the daemon, and the state of the daemon (active or inactive).

4. To get long status from the **hostmibd** daemon, enter:

   ```
   lssrc -ls hostmibd
   ```

   If you are the root user, this long form of the status report lists the configuration parameters in **/etc/hostmibd.conf**.

Files

/etc/hostmibd.conf	Defines the configuration parameters for **hostmibd** command.
/etc/mib.defs	Defines the Management Information Base (MIB) variables the SNMP agent and manager should recognize and handle.

Related Information

The **snmpdv3** daemon, **snmpmibd** daemon.

---

hostname Command

Purpose

Sets or displays the name of the current host system.

Syntax

/usr/bin/hostname [*HostName*] [**-s**]

Description

The **/usr/bin/hostname** command displays the name of the current host system. Only users with root user authority can set the host name. The **mkdev** command and the **chdev** commands also set the host name permanently. Use the **mkdev** command when you are defining the TCP/IP instance for the first time.

You can use the System application in Web-based System Manager (wsm) to change system characteristics. You could also use the System Management Interface Tool (SMIT) **smit mkhostname** fast path to run this command.

Flags

-s Trims any domain information from the printed name.

Parameters

HostName Sets the primary name of the host.

Note: You must have root user authority to use the *HostName* parameter.

Security

Attention RBAC users and Trusted AIX users: This command can perform privileged operations. Only privileged users can run privileged operations. For more information about authorizations and privileges, see Privileged Command Database in *Security*. For a list of privileges and the authorizations associated with this command, see the **/etc/security/privcmds** database file.

Related Information

The **chdev** command, **mkdev** command.

The **gethostname** subroutine, **sethostname** subroutine.

For information on installing the Web-based System Manager, see Chapter 2: Installation and System Requirements in *AIX Version 6.1 Web-based System Manager Administration Guide*.

TCP/IP name resolution in *Networks and communication management*.

---

hosts2ldif Command

Purpose

Creates an LDAP Data Interchange Format (LDIF) file from a hosts file.

Syntax

hosts2ldif [**-i** *InputFile*] [**-o** *OutputFile*] [**-s** *SearchBase*]

Description

The **/usr/sbin/hosts2ldif** command creates a LDAP Data Interchange Format (LDIF) file from **/etc/hosts** or another file that looks like **/etc/hosts**. With no flags, the **/etc/hosts** file is used to create the **/tmp/hosts.ldif** LDIF file using `cn=hosts` as the baseDN.

The LDIF file created by this command is compliant with SecureWay® Directory Schema and is used for setting up the **ldap** mechanism. The **ldap** mechanism is supported, but the use of the **nis_ldap** mechanism rather than the **ldap** mechanism is recommended.

Flags

-i *InputFile*	Specifies the hosts file used for input.
-o *OutputFile*	Specifies the LDIF file used for output.
-s *SearchBase*	Specifies the baseDN of the host table on the LDAP server.

Examples

1. To create **/home/ldifhosts** from the **/etc/hosts** file, type:
   ```
   hosts2ldif -o /home/ldifhosts
   ```
2. To create **/tmp/hosts.ldif** from the **/home/hosts.bak** file, type:
   ```
   hosts2ldif -i /home/hosts.bak
   ```
3. To create **/home/ldifhosts** from the **/etc/hosts** file using `cn=hosttab` as the baseDN, type:
   ```
   hosts2ldif -o /home/ldifhosts -s cn=hosttab
   ```

Files

/etc/hosts	Contains the Internet Protocol (IP) name and addresses of hosts on the local network.

Related Information

TCP/IP name resolution in the *Networks and communication management*.

hp Command

Purpose

Handles special functions for the HP2640- and HP2621-series terminals.

Syntax

hp [**-e**] [**-m** ...]

Description

The **hp** command reads standard input (usually output from the **nroff** command), and writes to standard output, which is usually Hewlett-Packard 2640- and 2621-series terminal displays.

If your terminal has the display enhancement feature, you can display subscript characters and superscript characters. With the mathematical-symbol feature, you can display Greek characters and other special characters, with two exceptions. The **hp** command approximates the logical operator NOT with a right arrow and shows only the top half of the integral sign.

Overstrike characters are characters followed by a backspace and another character. They appear underlined or in inverse video (depending on terminal enhancements) if either the overwritten character or the character typed after the backspace is an underscore character.

> **Note:** Some sequences of control characters (reverse line-feeds and backspaces) can make text disappear from the display. Tables with vertical lines generated by the **tbl** command may be missing lines of text containing the bottom of a vertical line. You may be able to avoid these problems by first piping the input through the **col** command and then through the **hp** command.

Flags

-e Shows overstruck characters underlined, superscript characters in half-bright, and subscript characters in half-bright underlined. Otherwise, all overstruck characters, subscript characters, and superscript characters appear in inverse video (dark-on-light). Use this flag only if your display has the display enhancements feature.

-m Produces only one blank line for any number of successive blank lines in the text.

Related Information

The **col** command, **eqn** command, **greek** command, **nroff** command, **tbl** command.

hplj Command

Purpose

Postprocesses the **troff** command output for the HP LaserJet Series printers.

Syntax

hplj [**-F** *Directory*] [**-quietly**] [**-landscape**] [*File* ...]

Description

The **hplj** command processes the output of the **troff** command for output to Hewlett-Packard LaserJet Series printers.

If given one or more files as options, the **hplj** command processes those files. If no files are specified, it acts as a filter interpreting standard input. The parameter *File* specifies files the **hplj** command processes to output on an HP Laser Jet Series printer.

> **Note:** The **hplj** command can use the K cartridge or Text-Equations cartridge if installed in the printer. (The Text-Equations cartridge, HP part number C2053A #C07, supersedes the K cartridge.) The default font files assume one of the cartridges is installed. If you do not have a K cartridge, use the downloaded bit-mapped fonts instead. To do this, run the **no_cart** shell script in the font directory for the HP printer (**/usr/lib/font/devhplj**).

Incorrect output can occur if your font files assume either cartridge is mounted when it is not. Incorrect output can also occur if other cartridges or soft fonts are installed, in addition to the K cartridge or Text-Equations cartridge.

The **hplj** command depends on the files with names ending in **.out** in the **/usr/lib/font/devhplj** file. This command does not produce reasonable output unless these files have been properly set up. See the **troff** font file format document for more information.

Flags

-F*Directory* Identifies the specified directory as the place to find the font file. By default, the **hplj** command looks for font files in the **/usr/lib/font/devhplj** directory.

-quietly Suppresses all nonfatal error messages.

-landscape Prints the specified file in landscape format. A landscape page is oriented so that for normal reading, the width of the page is greater than its length. By default, the **hplj** command prints in portrait orientation.

> **Note:** Landscape is only available in the Courier font on the Hewlett-Packard Jet II printer. Therefore, **troff** documents must be formatted in the Courier font. To accomplish this, insert the following lines at the beginning of the **troff** input file:
>
> ```
> .fp 1 C
> .fp 2 C
> .fp 3 CB
> ```
>
> The Courier font is loaded onto font positions #1 & #2 and Courier-Bold onto position #3.

Examples

1. To print a **troff** file named `foo` on the printer called `hp` using the **lp** command, enter:
   ```
   troff -mm -Thplj foo | hplj | lp -dhp -o -dp
   ```
2. To print a **troff** file named `boo` on printer called `hp` using the **qprt** command, enter:
   ```
   troff -mm -Thplj boo | hplj | qprt -dp -Php
   ```

> **Note:** The **-dp** flag in both examples sends the printer data to the print device in pass-through (unmodified) mode.

File

/usr/lib/font/devhpl/*.out Contains font files.

Related Information

The **troff** command formats text for printing on typesetting devices.

The **troff** font file format specifies description files for the troff command.

---

hpmcount Command

Purpose

Measures application performance.

Syntax

hpmcount [**-a**] [**-b** *time_base*] [**-d**] [**-D** *metrics*] [**-g** *event_groups*] [**-H**] [**-k**] [**-m** *metrics_groups*] [**-o** *file*] [**-s** *set*] [**-x**] *command*

hpmcount [**-h**]

Description

The **hpmcount** command provides the execution wall clock time, hardware performance counters information, derived hardware metrics, and resource utilization statistics (obtained from the **getrusage()** system call) for the application named by *command*.

Event types to be monitored and the associated hardware performance counters are specified by setting the **-s** option, by specifying an event group name, set number, or a comma-separated list of set numbers in the **HPM_EVENT_SET** environment variable, or by specifying counter/event pairs POWER3 / PowerPC 604 RISC Microprocessor) or an event group name (POWER4 and later) in the **libHPM_events** input file (takes precedence over **HPM_EVENT_SET**). Each set can be qualified by a counting mode. An event group number or name can be specified by setting the **-g** option or specifying a comma-separated list of event groups in the **HPM_EVENT_GROUP** environment variable. In the same manner, each event group can be qualified by a counting mode.

Valid event set numbers run from `1` to an upper limit dependent upon the processor type, which can be listed using the **pmlist** command. A comma-separated list of event sets can be specified instead of a set number, in which case the counter multiplexing mode is selected. To select all event sets, set the number value to `0`.

A comma-separated list of derived metrics can be specified by setting the **-D** option. Each derived metric can be qualified by a counting mode.

A list of derived metric groups can be specified by setting the **-m** option or by specifying a comma-separated list of derived metric groups in the **HPM_PMD_GROUP** environment variable. This allows the selection of all of the derived metrics pertaining to the specified groups. Each metric group can be qualified by a counting mode.

System and hypervisor (for processors supporting hypervisor mode) activity can be included in counting by specifying the **-k** and **-H** options.

When counting in the multiplexing mode, the results must be normalized to be used. The default base used for the data normalization is the timebase. The **-b** option allows for the use of the PURR time or the SPURR time (when supported by the processor) for the data normalization. The base for the data normalization can also be defined using the **HPM_NORMALIZE** environment variable.

Results can be output in XML format using the **-x** option.

Flags

-a Aggregates the counters on POE runs.

-b *time_base* Selects a base for the data normalization. The available bases are as follows:

time timebase

purr PURR time (when available)

spurr SPURR time (when available)

The default value is **time**.

-d Adds detailed set counts for counter multiplexing mode.

-D *metrics* Selects a list of derived metrics to be evaluated. Each derived metric can be qualified by a counting mode as follows:

```
metric:counting_modes
```

(See the **-m** option for available counting modes.)

-g *event_groups* Lists a predefined group of events or a comma-separated list of event group names or numbers. When a comma-separated list of groups is used, the counter multiplexing mode is selected. Each event group can be qualified by a counting mode as follows:

```
event_group:counting_modes
```

(See the **-m** option for available counting modes.)

-H Adds hypervisor activity on behalf of the process.

-h Displays help message.

-k Adds system activity on behalf of the process.

-m *metrics_groups* Selects a list of derived metric groups to be evaluated. The derived metric group refers to all derived metrics that do not belong to a specific derived metric group. Each metric group can be qualified by a counting mode as follows:

```
metric_group_name:counting_modes
```

The available counting modes are as follows:

u user mode

k kernel mode

h hypervisor mode

r runlatch mode

n nointerrupt mode

-o *file* Output file name.

-s *set* Lists a predefined set of events or a comma-separated list of sets (`1` to *N*, or `0` to select all. See the **pmlist** command.) When a comma-separated list of sets is used, the counter multiplexing mode is selected. Each set can be qualified by a counting mode as follows:

```
event_set:counting_modes
```

(See the **-m** option for available counting modes.)

-x Displays results in XML format.

Parameters

command Specifies the executed program for which performance measurements are made.

Environment Variables

The following environment variables directly affect the execution of the **hpmcount** command (there are additional **MP_*** environment variables that influence the execution of parallel programs).

HPM_EVENT_SET
Selects one of the event sets. The value can be an integer from 1 to 6 on POWER3 systems, 1 to 4 on PowerPC 604 RISC Microprocessor systems, or 1 to a processor-dependent upper limit on POWER4 and later systems. This environment variable is also used to select an event group name on POWER4 and later systems. A comma-separated list of event sets can be specified. In this case, the counter multiplexing mode is selected. Each event set can be qualified by a counting mode as follows:

```
event_set_number:counting_modes
```

The **-g** or **-s** option takes precedence over this variable. The **HPM_EVENT_GROUP** environment variable takes precedence over this variable.

HPM_EVENT_GROUP
Selects the event groups. A comma-separated list of event groups can be specified. In this case, the counter multiplexing mode is selected. Each event group can be qualified by a counting mode as follows:

```
event_group_number:counting_modes
```

The **-g** or **-s** option takes precedence over this variable. The **HPM_EVENT_GROUP** environment variable takes precedence over the **HPM_EVENT_SET** variable.

HPM_NORMALIZE
Provides the base to be used for the data normalization. The **-b** option takes precedence over this variable.

HPM_PMD_GROUP
Specifies a comma-separated list of derived metric groups. Each metric group can be qualified by a counting mode. The **-m** option takes precedence over this variable.

HPM_PMD_METRIC
Specifies a comma-separated list of derived metrics. Each derived metric can be qualified by a counting mode. The **-D** option takes precedence over this variable.

HPM_DIV_WEIGHT
Provides a weight (an integer greater than 1) to be used to compute weighted flips on POWER4 systems.

MP_CHILD
Used in a parallel environment when aggregate counts are specified to complement the output results file name (*myID*), synchronize collation of results, and identify verbose/debug diagnostic messages more closely.

MP_PROCS
The number of program tasks.

HPM_AGGREGATE_OUTPUT
Aggregates counts on POE applications (forces the command line argument **-a**). With this flag, a single file performance file is generated for all tasks. This only works with POE or Load Leveller, and it requires the availability of a parallel file system (such as GPFS) on the system.

HPM_LOG_DIR
When this flag is set, **hpmcount** writes a **hpm_log.***id* file with the performance data in the provided directory. This is in addition to the regular output.

MP_PARTITION
On POE applications, *id* is a POE ID provided by **MP_PARTITION**. Otherwise, it is the *pid*. Also names internal lock and data files.

HPM_MX_DURATION
When counting in counter multiplexing mode, this flag specifies the duration of each slice of time. It is expressed in ms, and must lie in the range of 10 ms - 30 s. When this flag is not set, the default value used for the time slice duration is 100 ms.

In addition, the following environment variables, supplied by the user, specify estimations of memory, cache, and TLB miss latencies for the computation of derived metrics. These environment variables do not take precedence over the same estimations eventually provided in the file **HPM_flags.env**, if present.

- **HPM_MEM_LATENCY**
- **HPM_L3_LATENCY**
- **HPM_L35_LATENCY**
- **HPM_AVG_L3_LATENCY**
- **HPM_AVG_L2_LATENCY**
- **HPM_L2_LATENCY**

- **HPM_L25_LATENCY**
- **HPM_L275_LATENCY**
- **HPM_L1_LATENCY**
- **HPM_TLB_LATENCY**

Exit Status

0	Successful completion.
>**0**	An error occurred.

Example

1. To run the **ls** command and write information concerning events in set 5 from hardware counters, enter:

   ```
   hpmcount -s 5 ls
   ```

2. To run the **ls** command and write information concerning events in sets 5, 2, and 9 from hardware counters using the counter multiplexing mode, enter:

   ```
   hpmcount -s 5,2,9 ls
   ```

3. To run the **ls** command and report derived metrics pertaining to the default and cpi_breakdown metric groups counted respectively in kernel+user+hypervisor mode and user mode, enter:

   ```
   hpmcount -m default:kuh,cpi_breakdown:u ls
   ```

Implementation Specifics

The **hpmcount** command uses the PMAPI thread-level API.

The **hpmcount** *command* parameter is not parsed as a command line for an application name with options. Instead, a shell script must be created that contains the command line.

Location

/usr/bin/perf/pmapi/hpmcount

Standard Input

Not used.

Standard Output

Performance monitoring results are written to **stdout**, unless the **-o** *file* option is specified on the command line.

Standard Error

Used only for diagnostic messages.

Files

The following input files are used if present.

libHPM_events

User-supplied event set file. This file does not take precedence over the command lines specified with the **-s** option. The format for a Power3/PowerPC 604 RISC Microprocessor counter/event pair is *counternumber eventname*. For example:

```
0 PM_LD_MISS_L2HIT
1 PM_TAG_BURSTRD_L2MISS
2 PM_TAG_ST_MISS_L2
3 PM_FPU0_DENORM
4 PM_LSU_IDLE
5 PM_LQ_FULL
6 PM_FPU_FMA
7 PM_FPU_IDLE
```

A comma-separated list of events can also be specified. This turns on the counter multiplexing mode:

```
0 PM_CYC,PM_FPU_FIN,PM_IC_MISS
1 PM_LD_CMPL,PM_INST_CMPL,PM_DC_MISS
2 PM_INST_CMPL,PM_FPU_WT,PM_INST_CMPL
3 PM_LD_MISS_DC_XU,PM_CYC,PM_CYC
```

For a POWER4 event group name, the format is *event_group_name*. For example:

```
pm_hpmcount1
```

A comma-separated list of events can also be specified. This turns on the counter multiplexing mode:

```
pm_hpmcount1,pm_hpmcount2,pm_basic
```

HPM_flags.env

File containing environment variable/value pairs used for the computation of derived metrics. For example:

```
HPM_L2_LATENCY 12
HPM_EVENT_SET  5
```

./.hpm_lockfile_mp_partition_

Lock file. This file is reserved for the **hpmcount** command's internal use.

./.hpm_datafile_mp_partition_

Accumulative results file. This file is reserved for the **hpmcount** command's internal use.

The following output files are used.

file_myID_**.**_pid_

File specified with the **-o** option for **hpmcount** output results, where *myID* is taken from the **MP_CHILD** environment variable, with a default value of `0000`.

HPM_LOG_DIR/hpm_log.MP_PARTITION or **HPM_LOG_DIR/hpm_log.***pid*

Log file specified for aggregate counters on POE runs.

./.hpm_lockfile_mp_partition_

Lock file. This file is reserved for the **hpmcount** command's internal use.

./.hpm_datafile_mp_partition_

Accumulative results file. This file is reserved for the **hpmcount** command's internal use.

Related Information

The "hpmstat Command" on page 706, the **pmlist** command.

The **getrusage** subroutine, the **pm_initialize** subroutine.

The Performance Monitor API Programming in *AIX Version 6.1 Performance Tools Guide and Reference*.

hpmstat Command

Purpose

Provides system-wide hardware performance counter information.

Syntax

hpmstat [**-b** *time_base*] [**-d**] [**-D** *metrics*] [**-g** *event_groups*] [**-H**] [**-k**] [**-m** *metrics_groups*] [**-o** *file*] [**-r**] [**-s** *set*] [**-T**] [**-U**] [**-u**] [**-x**] *interval count*

hpmstat [**-h**]

Description

The **hpmstat** command provides the execution wall clock time, hardware performance counters information, and derived hardware metrics. It can only be used by a user with root privilege.

When specified without command line options, **hpmstat** counts the default 1 iteration of user, kernel, and hypervisor (for processors supporting hypervisor mode) activity for 1 second for the default set 1 of events. It then writes the raw counter values and derived metrics to standard output. By default, **runlatch** is disabled so that counts can be performed while executing in idle cycle.

When the **-U** option is specified, *interval* is in microseconds, the iteration *count* is infinity, and derived metrics are not calculated and written to standard output. This option is ignored if the counter multiplexing mode is specified.

When the **-T** option is specified, output information is preceded by the time stamp (seconds plus microseconds) and timing information is written as time stamps instead of time in seconds.

Event types to be monitored and the associated hardware performance counters are specified using either the set **-s** option or by specifying an event group name or set number in the **HPM_EVENT_SET**one environment variable. Alternatively, specify counter/event pairs (POWER3 / PowerPC 604 RISC Microprocessor) or an event group name (POWER4 and later) in the **libHPM_events** input file (takes precedence over **HPM_EVENT_SET**). Each set can be qualified by a counting mode. An event group number or name can be specified by setting the **-g** option or specifying a comma-separated list of event groups in the **HPM_EVENT_GROUP** environment variable. In the same manner, each event group can be qualified by a counting mode.

A comma-separated list of event sets can be specified instead of a set number, in which case the counter multiplexing mode is selected. To select all event sets, set the set number value to 0.

Valid event set numbers run from 1 to an upper limit dependent upon the processor type, which can be listed using the **pmlist** command.

A comma-separated list of derived metrics can be specified by setting the **-D** option. Each derived metric can be qualified by a counting mode.

A list of derived metric groups can be specified by setting the **-m** option or by specifying a comma-separated list of derived metric groups in the **HPM_PMD_GROUP** environment variable. This allows to select all the derived metrics pertaining to the specified groups. Each metric group can be qualified by a counting mode.

When counting in the multiplexing mode, the results must be normalized to be used. The default base used for the data normalization is the timebase. The **-b** option allows for the use of the PURR time or the

SPURR time (when supported by the processor) for the data normalization. The base for the data normalization can also be defined using the **HPM_NORMALIZE** environment variable.

Results can be output in XML format using the **-x** option.

Flags

-b *time_base*
Selects a base for the data normalization. The available bases are as follows:

time timebase

purr PURR time (when available)

spurr SPURR time (when available)

The default value is **time**.

-d
Adds detailed set counts for counter multiplexing mode.

-D *metrics*
Selects a list of derived metrics to be evaluated. Each derived metric can be qualified by a counting mode as follows:

```
metric:counting_modes
```

(See the **-m** option for available counting modes.)

-g *event_groups*
Lists a predefined group of events or a comma-separated list of event group names or numbers. When a comma-separated list of groups is used, the counter multiplexing mode is selected. Each event group can be qualified by a counting mode as follows:

```
event_group:counting_modes
```

(See the **-m** option for available counting modes.)

-H
Counts hypervisor activity only.

-h
Displays help message.

-k
Counts system activity only.

-m *metrics_groups*
Selects a list of derived metric groups to be evaluated. The default derived metric group refers to all derived metrics that do not belong to a specific derived metric group. Each metric group can be qualified by a counting mode as follows:

```
metric_group_name:counting_modes
```

The available counting modes are as follows:

u user mode

k kernel mode

h hypervisor mode

r runlatch mode

n nointerrupt mode

-o *file*
Output file name.

-r
Enables **runlatch** and disables counts while executing in idle cycle.

-s *set*
Lists a predefined set of events or a comma-separated list of sets (`1` to *N*, or `0` to select all. See the **pmlist** command.) When a comma-separated list of sets is used, the counter multiplexing mode is selected. Each set can be qualified by a counting mode as follows:

```
event_set:counting_modes
```

(See the **-m** option for available counting modes.)

-T
Writes time stamps instead of time in seconds.

-U Puts counting time interval in microseconds. This option is ignored if the counter multiplexing mode is specified.

-u Counts user activity only.

-x Displays results in XML format.

Parameters

interval Displays the counting time interval in seconds or microseconds, with a default value of 1.

count Shows the number of iterations to count. The default is 1 with an interval in seconds, and infinity when the option **-U** is specified.

Environment Variables

The following environment variables directly affect the execution of the **hpmstat** command (there are additional **MP_*** environment variables that influence the execution of parallel programs).

HPM_EVENT_SET Selects one of the event sets. The value can be an integer from 1 to 6 on POWER3 systems, 1 to 4 on PowerPC 604 RISC Microprocessor systems, or 1 to a processor-dependent upper limit on POWER4 and later systems. This environment variable is also used to select an event group name on POWER4 and later systems. Each event set can be qualified by a counting mode as follows:

```
event_set_number:counting_modes
```

The **-g** or **-s** option takes precedence over this variable. The **HPM_EVENT_GROUP** environment variable takes precedence over this variable.

HPM_EVENT_GROUP Selects the event groups. A comma-separated list of event groups can be specified. In this case, the counter multiplexing mode is selected. Each event group can be qualified by a counting mode as follows:

```
event_group_number:counting_modes
```

The **-g** or **-s** option takes precedence over this variable. The **HPM_EVENT_GROUP** environment variable takes precedence over the **HPM_EVENT_SET** variable.

HPM_NORMALIZE Provides the base to be used for the data normalization. The **-b** option takes precedence over this variable.

HPM_PMD_GROUP Specifies a comma-separated list of derived metric groups. Each metric group can be qualified by a counting mode. The **-m** option takes precedence over this variable.

HPM_PMD_METRIC Specifies a comma-separated list of derived metrics. Each derived metric can be qualified by a counting mode. The **-D** option takes precedence over this variable.

HPM_DIV_WEIGHT Provides a weight (an integer greater than 1) to be used to compute weighted flips on POWER4 systems.

HPM_MX_DURATION When counting in counter multiplexing mode, this flag specifies the duration of each slice of time. It is expressed in ms, and must lie in the range of 10 ms - 30 s. When this flag is not set, the default value used for the time slice duration is 100 ms.

In addition, the following environment variables, supplied by the user, specify estimations of memory, cache, and TLB miss latencies for the computation of derived metrics. These environment variables do not take precedence over the same estimations eventually provided in the file **HPM_flags.env**, if present.

- **HPM_MEM_LATENCY**
- **HPM_L3_LATENCY**
- **HPM_L35_LATENCY**
- **HPM_AVG_L3_LATENCY**
- **HPM_AVG_L2_LATENCY**
- **HPM_L2_LATENCY**
- **HPM_L25_LATENCY**
- **HPM_L275_LATENCY**
- **HPM_L1_LATENCY**
- **HPM_TLB_LATENCY**

Exit Status

0	Successful completion.
>**0**	An error occurred.

Example

To write information for system, user, and hypervisor activity over a 1 second interval concerning events in set 2 from hardware counters, type:

```
hpmstat -s 2
```

Implementation Specifics

The **hpmstat** command uses the PMAPI system-level API. Because the system-level APIs report bogus data if the thread-level API is in use, system-level API calls are not allowed at the same time as thread-level API calls. The allocation of a thread context will take the system-level API lock, which will not be released until the last context has been deallocated. Because of this, **hpmstat** counts will not be accurate if a program measured by **libhpm** or **hpmcount** is activated during the time that **hpmstat** is active.

Location

/usr/bin/perf/pmapi/hpmstat

Standard Input

Not used.

Standard Output

Performance monitoring results are written to **stdout**, unless the **-o** *file* option is specified on the command line.

Standard Error

Used only for diagnostic messages.

Files

The following input files are used if present.

libHPM_events

User-supplied event set file. This file does not take precedence over the command lines specified with the **-s** option. The format for a Power3/PowerPC 604 RISC Microprocessor counter/event pair is *counternumber eventname*. For example:

```
0 PM_LD_MISS_L2HIT
1 PM_TAG_BURSTRD_L2MISS
2 PM_TAG_ST_MISS_L2
3 PM_FPU0_DENORM
4 PM_LSU_IDLE
5 PM_LQ_FULL
6 PM_FPU_FMA
7 PM_FPU_IDLE
```

For a POWER4 event group name, the format is *event_group_name*. For example:

```
pm_hpmcount1
```

HPM_flags.env

File containing environment variable/value pairs used for the computation of derived metrics. For example:

```
HPM_L2_LATENCY 12
HPM_EVENT_SET  5
```

The following output files are used.

file

File specified with the **-o** option for **hpmstat** output results.

Related Information

The **hpmcount** command, the **pmlist** command.

The **pm_initialize** subroutine.

The Performance Monitor API Programming in *AIX Version 6.1 Performance Tools Guide and Reference*.

---

hps_dump Command

Purpose

Dumps contents of Network Terminal Accelerator (NTX) adapter memory to a host file. This command only applies to AIX 4.2.1 or later.

Syntax

hps_dump [**-f** *Name*] [**-d** *Device*]

Description

The **hps_dump** command uses the loader interface to upload all of the memory from the adapter board into a file. This produces a snapshot of a system for later analysis and debugging. The first 1024 bytes of the file contains the following:

80 Identification string, includes version.
80 Time and date of dump from host system.

80 Comments.
268 Log table from the host adapter.
32 System address table.
8 Starting and ending address range of dump.
476 Padding to 1024 bytes total.

Flags

-f *Name* Specifies the name of the dump. Use this option to override the default filename **./hpscore**.
-d *Device* Specifies the raw device file name of the adapter. Use this option to override the default device name **/dev/rhp0**.

Exit Status

This command returns the following exit values:

0 Successful completion.
>0 An error occurred.

Security

Access Control: You must have root authority to run this command.

Auditing Events: N/A

Examples

1. To get a dump of memory of the default adapter to the file **hpscore** in the current directory, enter:
   ```
   hps_dump
   ```
2. To get a dump of memory of the default adapter to the file **hpsdebug** in the current directory of the default adapter, enter:
   ```
   hps_dump -f hpsdebug
   ```
3. To get a dump of memory of the adapter **/dev/rhp1** to the file **hpsdebug** in the current directory of the default adapter, enter:
   ```
   hps_dump -f hpsdebug -d /dev/rhp1
   ```

Files

/usr/bin/hps_dump Contains the **hps_dump** command.
/dev/rhp0 Default NTX raw device file name.

Related Information

The **/dev/rhp** file.

---

htable Command

Purpose

Converts host files to the format used by network library routines.

Syntax

/usr/sbin/htable [**-c** *connected-nets*] [**-l** *local-nets*] *input-file*

Note: Do not put a space on either side of the comma.

Description

The **htable** command converts host files in the format specified in RFC 810 to the format used by the network library routines. The conversion creates three files: the **/etc/hosts** file, the **/etc/networks** file, and the **/etc/gateways** file.

The **gethostbyname** subroutine uses the **hosts** file for mapping host names to addresses when the **named** daemon is not used. The **getnetent** subroutine uses the **networks** file for mapping network names to numbers.

The **gateways** file may be used by the **routed** daemon in identifying passive Internet gateways.

If any local **hosts**, **networks**, or **gateways** files (**localhosts**, **localnetworks**, or **localgateways** respectively) exist in the current directory, that file's contents are prepended to the output file. Of these, the **htable** program only interprets the **gateways** file. Prepending the contents allows sites to maintain local entries that are not normally present in the master database.

Flags

-c *connected-nets*	Specifies a list of networks to which the host is directly connected if the network routing daemons use the **gateways** file. Separate the networks with commas, and use the network name or standard Internet dot notation (for example, `-c arpanet,128.32,LocalEthernet`). The **htable** command only includes gateways that are directly connected to one of the networks specified or that can be reached from another gateway on a connected network.
-l *local-nets*	Specifies a list of networks for the **htable** command to treat as local. Take information about hosts on local networks only from the **localhosts** file. Separate the networks with commas, and use the network name or standard Internet dot notation (for example, `-l 128.32,local-ether-net`). Entries for local hosts from the main database are omitted so that the **localhosts** file can override entries in the input file (the file you specify on the command line).

Files

*/CurrentDirectory/***localgateways**	Contains local gateway information.
*/CurrentDirectory/***localhosts**	Contains local host name information.
*/CurrentDirectory/***localnetworks**	Contains local network information.

Related Information

The **gettable** command.

The **named** daemon, **routed** daemon.

The **gateways** file format, **hosts** file format, **networks** file format.

TCP/IP routing gateways in *Networks and communication management*.

hty_load Command

Purpose

Displays or downloads Network Terminal Accelerator (NTX) adapter configurations.

Syntax

hty_load [**-d** *Device*] [**-f** *ConfigFileName*]

Description

The **hty_load** command displays or downloads adapter configurations. If you issue this command without any flags, the system displays the current adapter configuration for the **/dev/rhp0** device file. Given a *Device* parameter, the **hty_load** command loads a configuration file into the tty driver. The tty driver uses the file to configure both the host presentation services (HPS) and the adapters.

Typically, the **hty_load** command is invoked from the **/etc/rc.ntx** file.

The Configuration File

The **hty_load** command uses a single configuration file to configure the adapters. Each entry is on a separate line. Entries are separated by new-line characters. Fields in an entry are separated by tabs or space characters. Entries in the configuration file have the following fields.:

MinorNumber Cluster NumberOfPorts

These fields have the following values:

MinorNumber Specifies the board's minor device number.

Cluster This field is always 1.

NumberOfPorts Specifies the number of hty devices. The number depends on the model of adapter you are using. The number of available channels is from 1 to 256 for a 2MB board or from 1 to 2048 for an 8MB board.

The configuration file also supports comments. Comment lines begin with a # (pound sign). Everything to the right of the comment character is ignored. Comment lines end with new-line characters.

Flags

-d *Device* Specifies the raw device file name of the adapter. Use this option to override the default device name **/dev/rhp0**.

-f *ConfigFileName* Specifies the driver configuration file name. The default configuration file is the **/etc/hty_config** file.

Exit Status

This command returns the following exit values:

0 Successful completion.

>0 An error occurred.

Security

Access Control: You must have root authority to run this command.

Auditing Events: N/A

Examples

To load the system configuration and use the default driver configuration file, enter:

```
hty_load -d /dev/rhp0
```

Files

/usr/bin/hty_load	Contains the **hty_load** command.
/etc/rc.ntx	Invokes the **hty_load** command.
/etc/hty_config	Default NTX driver configuration file name.
/dev/rhp0	Default NTX raw device file name.

Related Information

The **/dev/rhp** file.

---

hyphen Command

Purpose

Finds hyphenated words.

Syntax

hyphen [*File* ...]

Description

The **hyphen** command reads one or more English-language files, finds all the lines ending with hyphenated words, and writes those words to standard output. The parameter *File* specifies English-language files to be read by the **hyphen** command. The default is standard input. If no file is specified or if the - (hyphen) is specified as the last file name, the **hyphen** command reads standard input. The **hyphen** command can be used as a filter.

> **Note:** The **hyphen** command cannot read hyphenated words that are italic or underlined. The **hyphen** command sometimes gives unnecessary output.

Examples

To check the hyphenation performed by a text-formatting program on a file, enter:

```
mm [Flag...] [File...] | hyphen
```

Related Information

The **mm** command, **troff** command.

Appendix. Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described 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
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

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

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

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
Dept. LRAS/Bldg. 003
11400 Burnet Road
Austin, TX 78758-3498
U.S.A.

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

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

© Copyright IBM Corp. 1997, 2007

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Trademarks

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

AIX
AIX 5L
CICS
CICS/ESA
CICS/MVS
CICS/VSE
GPFS
HACMP
IBM

LoadLeveler
MVS
MVS/SP
MVS/XA
OS/400
POWER
PowerPC
RACF
RS/6000
SecureWay
SP
VSE/ESA
WebExplorer

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java and all Java-based trademarks and logos are registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Other company, product, or service names may be the trademarks or service marks of others.

Index

Special characters

/etc/qconfig file
  converting into /etc/qconfig.bin file
    using /user/lpd/digest command 125
/user/lpd/digest command 125
Empty 186, 207, 208

A

accounting system
  changing record formats 573
  starting 176
acct/* commands
  dodisk 176
adapter configuration
  displaying and downloading 713
altscreen command 195
arguments
  writing to standard output 268
arithmetic
  desk calculator 66
  factoring numbers 434

C

command history files 435
command lines
  parsing
    flags 629
    parameters 629
command path names 687
commands
  dd 68
  defvsd 78
  detachrset 84
  diff 115
  disable 128
  dosread 183
  ed 270
  edquota 309
  elogevent 329
  enotifyevent 342
  enscript 352
  env 363
  event response resource manager (ERRM)
    elogevent 329
    enotifyevent 342
    ewallevent 405
  ewallevent 405
  ex 407
  extendlv 428
  fccheck 438
  fcclear 439
  fcdecode 442
  fcdispfid 443
  fcfilter 444
  fcinit 446
commands *(continued)*
  fclogerr 449
  fcpushstk 456
  fcreport 462
  fcstkrpt 466
  fcteststk 468
  fencevsd 479
  find 498
  get 607
  getconf 616
  grpsvcsctrl 657
  ha_vsd 665
  ha.vsd 661
  haemqvar 667
  haemtrcoff 671
  haemtrcon 673
  haemunlkrm 675
  hagsns 679
  hagsvote 681
  hatsoptions 686
  hostent 692
  red 270
communication channel
  implementing 352
comparing
  text files 115
configuration variable values
  to standard output, writing
    using getconf command 616
control scripts
  grpsvcsctrl 657

D

dacinet command 1
daemon
  starting error logging 382
  terminating the error logging 396
daemons
  dhcprd 102
  dhcpsd 105
  fingerd 509
  ftpd 564
  glbd 638
  gssd 660
  haemd 666
  hagsd 677
date command 4
dbts command 8
dbx
  stophwp 49
  tracehwp 56
dbx <?>subcommands
  ? 15
  / 15
  addcmd 15
  alias 16
  assign 16

© Copyright IBM Corp. 1997, 2007

dbx <?>subcommands *(continued)*
attribute
thread attributes 17
call 18
case 18
catch 19
cleari 19
cont 20
corefile 21
delcmd 21
delete 21
detach 22
down 24
dump 24
edit 25
file 26
frame 26
func 26
goto 27
ignore 28
list 29
malloc 31
map 31
move 32
multproc 33
mutex
thread debugging 34
next 35
nexti 35
prompt 37
rerun 38
return 39
run 40
screen 40
set 41
sh 45
skip 45
source 45
status 45
step 46
stepi 47
stop 47
stopi 50
thread
thread debugging 50
tls 54
tnext 54
tnexti 54
trace 55
tracei 57
tskip 57
tstep 58
tstepi 58
tstop 59
tstophwp 59
tstopi 60
ttrace 61
ttracehwp 62
ttracei 61
unalias 62
unset 63
dbx <?>subcommands *(continued)*
up 63
use 63
whatis 64
where 65
whereis 65
which 65
dbx command
aliases
creating 16
removing 62
application program
continuing 20
continuing execution 22
listing instructions from 30
removing traces and stops 21
application programs
continuing from the current stopping point 45
displaying component declarations of 64
running to a specified procedure 39
running to next machine instruction 35
running to the next source line 35
starting 40
starting an application 38
stopping 47
breakpoint stop
setting 59, 60
breakpoints
removing 19
changing functions 26
command prompt, changing 37
dbx program
stopping 38
description of 9
directories
search list, setting 63
editor
starting 25
expressions
printing the value of 36
function
changing to specified procedure or function 26
functions
current 24, 63
list of active 65
identifier
displaying full qualification of 65
lines
changing displayed 32
load characteristics
displaying 31
machine instructions
running single 47
multiprocess debugging 33
object code
running 18
procedures
list of active 65
running and printing 36
program counter address
changing 27

dbx command *(continued)*
register values
displaying 38
shell
passing commands to 45
signal trapping 19
stopping 28
source files
changing to specified files 26
displaying lines 29
searching backward in 15
searching forwards in 15
source lines
running single 46
running specified 27
stop subcommand
displaying 45
stopping the dbx program 38
stops
removing at a source line 19
setting at a specified location 50
subcommands
handler 27
kthread 29
onceblock 36
printing list of 27
system symbols
changing interpretation of 18
displaying full qualifications 65
thread debugging 17, 19, 34, 50
trace subcommand
displaying 45
tracing
information, printing 55
turning on 57, 61
tracing information
printing 61
variables
assigning values to 16
defining values for 41
deleting 63
displaying 24
virtual terminals, opening 40
watchpoint stops
setting 59
watchpoint traces
setting 62
dbx subcommands
clear 19
condition
thread debugging 19
gotoi 27
handler 27
help 27
kthread 29
listi 30
onceblock 36
plugin 36
pluginload 36
pluginunload 36
print 35
dbx subcommands *(continued)*
quit 37
registers 38
dd command 68
debugging programs 9
defif method 73
definet method 74
defragmented file system 75
defvsd command 78
deleteX11input command 80
delta files
creating 81
deroff command 83
detachrset command 84
devices
installing software support 85
naming a 87
devinstall command 85
devnm command 87
df command 87
dfmounts command 92
dfpd command 93
dfsck command 94
dfshares command 96
DHCP 105
dhcpaction command 97
dhcpcd daemon 98
dhcpcd6 command 100
dhcprd daemon 102
dhcpsconf command 103
dhcpsd daemon 105
dhcpsdv6 daemon 106
diag command 108
diaggetrto command 111
diagnostics
hardware 108, 112
diagrpt command 112
diagsetrto command 113
diction command
description of 114
diff command 115
diff3 command 118
diffmk command 120
dig 121
directories
comparing two 126
DOS files
listing 179
dirname command 127
disable
printer queue
using disable command 128
disable command 128
disk accounting
generating data by user ID 130
disk map
printing information on 267
disk usage 257
diskettes
copying 511

diskettes *(continued)*
formatting
fdformat command 472
format command 520
diskusg command 130
dispgid command 132
dispuid command 133
dist command 134
dmadm command 137
dmf command 138
verbs
add_to 153
check_adm 166
check_adm_serv 166
check_serv 166
clear 149
create 141
destroy 143
enumerate 145
master 158
mount 162
place 159
remove_from 155
repair 164
resolve 164
set 149
show 147
source 158
unmount 163
unplace 160
update 158
validate 164
dmpuncompress command 167
dms command 168
dms_enable_fs command 171
dnssec-keygen 171
dnssec-makekeyset 173
dnssec-signkey 174
dnssec-signzone 175
dodisk command 176
domainname command 178
don055101 4
DOS
formatting diskettes 181
DOS files
copying to 184
copying to AIX 183
deleting 178
directory for
listing 179
dosread command 183
dp command 186
dpid2 Daemon 186
drm_admin command 188
drmgr command 191
dslpaccept command 196
dslpaccess command 197
dslpadmin command 198
dslpdisable command 202
dslpenable command 203
dslpprotocol command 203
dslpreject command 205
dslpsearch command 206
dspcat command 207
dspmsg command 208
dtaction command 210
dtappintegrate command 212
dtlogin command 213
dtscript 239
dtsession command 240
du command 257
dump command 259
dumpfs command 267
disk map 267
i node map 267
superblock 267
dynamic host configuration protocol
forwarding bootp and dhcp packets
dhcprd daemon 102
graphical user interface
dhcpsconf command 103
run NIM and DHCP concurrently
bootptodhcp command 97
serve address and configuration information
dhcpcd daemon 98
dhcpsd daemon 105
updates the DNS server
dhcpaction command 97
dynamic logical partitioning
drmgr command 191

E

echo command 268
ed command 270
ed editor
adding text 276
capabilities of 276
changing text 279
command mode 270
copying text 280
deleting text 281
displaying text 286
joining lines 288
making global changes 290
marking text 291
moving text 291
saving text 292
searching text 293, 294
splitting lines 288
text input mode 270
undoing changes 297
edit command 302
edit editor
adding text 305
addresses
types 303
addressing lines in a file 303
changing
current file name 305
text 305
command mode 302

edit editor *(continued)*
copying text 308
current line
finding out 306
deleting text 306
displaying
current file name 306
current file status 306
text 306
editing additional files 307
ending 307
exiting 307
file name
changing 305
displaying 306
file status
displaying 306
global changes, making 307
modes of operation
command mode 302
text input mode 302
moving text 308
saving
files after system crash 308
text 309
subcommands
using 304
substituting text 309
text input mode 302
undoing changes 309
edit text
by line
using ed command 270
editing
user and group quotas
using edquota command 309
editing lines
interactively
using ex command 407
edquota command 309
efsenable command 311
efskeymgr command 313
efsmgr command 317
egrep command 319
eimadmin command 321
elogevent command 329
elogevent script 329
emgr command 331
emsvcsctrl script 337
enable command 340
enotifyevent command 342
enotifyevent script 342
enq command 343
enroll command 352
enscript command 352
env command 363
environment
displaying current 363
environment, sets
for command execution
using env command 363
epkg command 365
eqn command
removing command constructs from 83
errclear command 374
errctrl command 376
errdead command 381
errdemon daemon 382
errinstall command 384
errlogger command 387
ERRM
event information
logging 329
ERRM commands
elogevent 329, 405
enotifyevent 342
ERRM scripts
elogevent 329, 342
ewallevent 405
errmsg command 388
error log
creating an entry for an operator 387
deleting entries from 374
processing a report of logged 390
errpt command 390
errstop command 396
ethchan_config command 404
event information
logging 329
event response resource manager (ERRM)
commands
elogevent 329
enotifyevent 342
ewallevent 405
event information
logging 329
scripts
elogevent 329
enotifyevent 342
ewallevent 405
ewallevent command 405
ewallevent script 405
ex command 407
execerror command 408
execrset command 409
execution of command
sets environment
using env command 363
execution profiles
producing 641
expand command 410
explain command 413
explore command 413
exportfs 414
exportvg command 422
expr command 423
expressions
evaluating 423
finding files with matching
using find command 498
extendlv command 428
extendvg command 430

F

f command 432
factor command 434
fasthalt command 683
fc command 435
fcstat command 463
fdformat command 472
fencevsd command 479
ff command 481
fg command 483
file
  displaying number of blocks 257
  enqueuing 343
  marking difference 120
  searching for a pattern
    using egrep command 319
    using grep command 650
file command 486
file processes
  listing 571
file system
  checking for consistency
    using dfsck command 94
    using fsck command 533
  conducting interactive repairs
    using dfsck command 94
    using fsck command 533
  debugging 538
  listing file names 481
  listing statistics 481
  reporting information on space 87
file systems
  defragmented 75
file types
  determining 486
filemon command 488
fileplace command
  fileblock placement 496
files 178, 486
  comparing 126
    text 115
    three 118
  converting and copying 68
  copying
    from DOS 183
    to DOS 184
  creating a specified version of SCCS
    using get command 607
  deleting
    DOS 178
  displaying
    first few lines 688
  finding with matching expressions
    using find command 498
  transferring between local and a remote host 555, 557, 559
  type
    determining 486
find command 498
finger command 506
  example of 433, 508
fingerd daemon 509
flags
  parsing 629
flcopy command 511
flush-secldapclntd 512
fmt command 513
fold command 514
folder command 515
folders
  listing 515
  listing in mail directory 518
  selecting 515
folding lines for output device 514
foreground jobs 483
format command 520
FORTRAN
  splitting into separate files 549
fortune command 522
forw command 523
fpm command 526
FRCA
  controlling and configuring 530
frcactrl command 530
from command 532
fsck command 533
fsdb command 538
fsplit 549
ftp command 550
ftpd daemon
  description of 564
  file transfer protocol requests 567
  subtree guidelines 567
fuser command 571
fwtmp command 573
fxfer command 574

G

games
  fortune 522
  go fish 510
  hangman 685
gated daemon
  description of 586
  manipulating with SRC 587
  signals 586
gencat command 591
gencore command 594
genfilt command
  adding filter rules 594
genkex command 599
genkld command
  shared objects list 599
genld command
  loaded objects list 600
gensyms command 601
genxlt command 605
get command 607
getconf command 616
getdev command 623
getdgrp command 625

getea command 628
getopt command 629
getrunmode 632
getsecconf 633
getsyslab 634
gettable command 635
gettrc command 635
getty command 636
glbd (global location broker daemon)
  description of 638
gprof command 641
grap command 646
graphs
  typesetting 646
greek command 650
grep command 650
group quotas
  editing
    using edquota command 309
groups
  displaying membership of a 653
  verifying the definition of 654
groups command 653
grpck command 654
grpsvcsctrl command 657
gssd 660

H

ha_vsd command 665
ha.vsd command 661
haemd daemon 666
haemd_HACMP program 666
haemqvar command 667
haemtrcoff command 671
haemtrcon command 673
haemunlkrm command 675
hagsd daemon 677
hagsns command 679
hagsvote command 681
halt command 683
hangman command 685
hash command 687
hatsoptions command 686
HCON
  files
    transferring between local and host system 574
head command 688
help
  displaying information 689
history files 435
hlpdhcpcd 98
hlpdhcprd 102
hlpdhcpsd 105
hlpecho 268
hlpedit 302
hlpexplore 413
hlpfactor 434
hlpfile 486
hlpfortune 522
hlpfsplit 549
hlpgprof 641
hlphangman 685
hlpregisters 38
host command 690
host name
  resolving into Internet address 690
hostent command 692
hostid command 694
hostmibd daemon 695
hostname command 697
hp command 698, 699
HP LaserJet series II printer
  postprocessing troff command output 699
HP2621-series terminals
  setting special functions 698, 699
HP2640-series terminals
  setting special functions 698, 699
hplj command 699
hpmcount command 701
hpmstat command 706
hps_dump command 710
htable command 711
hty_load command 713
hyphen
  finding words with 714
hyphen command 714

I

i node map
  printing information on 267
input extension record
  deleting 80
integer arithmetic 423
Internet address
  resolving into a host name 690

J

job control 483

K

kernel extension lists 599

L

logical volume
  increasing size with PP
    using extendlv command 428

M

mail
  determining the origin of 532
  formatting messages prior to sending 513
matching expressions
  finding files with
    using find command 498

message catalog
creating 591
displaying 207
displaying a message 208
modifying 591
message facility commands
dspcat 207
dspmsg 208
gencat 591
messages
adding to the error logging message catalog 388
forwarding
forw command 523
installing in error logging message sets 384
listing 515
mail directory 518
redistributing 134
selecting 515
MH
dp command 186
monitoring performance
file system performance 488
Multiple Screen utility
starting of 195

N

NCS daemons
glbd 638
NIS commands
domainname 178
nroff command
removing command constructs from 83
NTX commands
hps_dump 710
hty_load 713

O

object file
dumping selected parts 259
output
converting from Teletype Model 37 650
writing to specified path 127

P

parameters
parsing 629
path names 687
physical partitions
to increase size of LV
using extendlv command 428
pic command
processing graphs 646
ports
setting the characteristics of 636
PostScript
converting to text format
using enscript command 352
printer queue
disabling
using disable command 128
enabling 340
process accounting
writing messages to standard error 408
processor
halting
using fasthalt command 683
using halt command 683
programs
haemd_HACMP 666

R

red command 270
remote system
looking up users 433, 507

S

SCCS
delta files
creating 81
file, creating specified version of
using get command 607
SCCS commands
delta 81
get 607
scripts
elogevent 329
emsvcsctrl 337
enotifyevent 342
event response resource manager (ERRM)
elogevent 329
enotifyevent 342
ewallevent 405
ewallevent 405
grpsvcsctrl 657
server function for FTP protocol
TCP/IP
using ftpd daemon 564
shell scripts
parsing command-line arguments 630
standard output
writing character strings 268
writing system configuration variable values to
using getconf command 616
superblock
printing information on 267
system configuration variable values
to standard output, writing
using getconf command 616
system dump
extracting error records from 381
System V print subsystem
directory enabled printing
dslpaccept command 196, 206
dslpaccess command 197
dslpadmin command 198
dslpdisable command 202

System V print subsystem *(continued)*
directory enabled printing *(continued)*
dslpenable command 203
dslpprotocol command 203
dslpreject command 205

T

tbl command
removing command constructs from 83
TCP/IP
configuration database
controlling address-mapping entries 692
gateway routing functions
providing 586
host file
convert network library format 711
hosts
displaying name 697
getting ID 694
setting ID 694
setting name 697
inet instance
defining 74
instances
defining a network interface 73
NIC host table
obtaining 635
server function for FTP protocol
using ftpd daemon 564
TCP/IP commands
gettable 635
hostent 692
hostid 694
hostname 697
htable 711
TCP/IP daemons
fingerd 509
ftpd 564
gated 586
TCP/IP methods
defif 73
definet 74
TCP/IP smit commands
hostent command 692
Teletype Model 37 workstation
converting output from 650
terminals 698, 699
text
converting to PostScript format
using enscript command 352
text, edit
by line
using ed command 270
thesaurus
providing an interactive 413
time management
setting date and time 4
translation table
creating for the axeb command 605
creating for the ebxa command 605
troff command
removing command constructs from 83

U

user
showing information on 432, 506
user quotas
editing
using edquota command 309
users
providing help information 689

V

volume group
adding physical volumes 430
exporting definition from a set of physical volumes 422

W

WebExplorer
open main window
explore command 413
writing
and changing tabs to spaces 410

Readers' Comments — We'd Like to Hear from You

AIX Version 6.1
Commands Reference, Volume 2, d - h

Publication No. SC23-5244-00

We appreciate your comments about this publication. Please comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.

For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state on this form.

Comments:

Thank you for your support.

Submit your comments using one of these channels:

- Send your comments to the address on the reverse side of this form.
- Send your comments via e-mail to: aix6koub@austin.ibm.com

If you would like a response from IBM, please fill in the following information:

Name	Address
Company or Organization	
Phone No.	E-mail address

Readers' Comments — We'd Like to Hear from You
SC23-5244-00

IBM

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Information Development
Department 04XA-905-6C006
11501 Burnet Road
Austin, TX 78758-3493

Fold and Tape | **Please do not staple** | Fold and Tape

Cut or Fold
Along Line

Printed in U.S.A.

SC23-5244-00