HP-UX Reference

Release 11i

System Administration Commands

Section 1M

Part 2 of 2 (N-Z)

Volume 4 of 9

Edition 1

Customer Order Number: B2355-90688

Manufacturing Part Number: B2355-90692
E1200

Printed in: United States

Legal Notices

Warranty

Restricted Rights Legend

Copyright Notices

Trademark Notices

Publication History

The manual publication date and part number indicate its current edition. The printing date will change when a new edition is printed. Minor changes may be made at reprint without changing the printing date. The manual part number will change when extensive changes are made.

Manual updates may be issued between editions to correct errors or document product changes. To ensure that you receive the updated or new editions, you should subscribe to the appropriate product support service. See your HP sales representative for details.

First Edition: December 2000 (HP-UX Release 11i)

Volume Four
Table of Contents

Section 1M
Part 2

Volume Four
Table of Contents

Section 1M
Part 2

Table of Contents
Volumes Three and Four

Section 1M: System Administration Commands

Table of Contents
Volumes Three and Four

Table of Contents
Volumes Three and Four

Table of Contents
Volumes Three and Four

Notes

Section 1M
Part 2

System Administration Commands
(N-Z)

Section 1M
Part 2

System Administration Commands
(N-Z)

NAME

naaagt - Native Agent Adapter (NAA)

SYNOPSIS

export HP_NAA_CNF=*naaCnf*

export HP_NAA_PORT=*snmpPort*

export HP_NAA_GET_COMMUNITY=*community*

/usr/sbin/naaagt [**-K**] [**-n**] [**-E** *priority*] [**-m** *logMask*] ...

/usr/sbin/naaagt { **-h** | **-help** }

DESCRIPTION

The Native Agent Adapter (**naaagt**) allows third-party SNMP agents to work with the HP SNMP Master Agent (**snmpdm**).

The Native Agent Adapter runs as a subagent to the HP SNMP Master Agent. **naaagt** reads the **naaCnf** file (see the **HP_NAA_CNF** environment variable, described below), and it registers each object identifier (OID) with **snmpdm**. See the *naaCnf File Format* section.

After registration is complete, **naaagt** receives SNMP requests from **snmpdm** and forwards them to a non-standard UDP port on the same system (see the **HP_NAA_PORT** environment variable, described below). **naaagt** also receives SNMP responses from the third-party SNMP agent, which is listening to the non-standard UDP port, and **naaagt** forwards the responses to **snmpdm**.

The third-party SNMP agent must listen to the same non-standard UDP port that **naaagt** sends to (**HP_NAA_PORT**), not the standard SNMP port number (161). Refer to third-party documentation for instructions on how to accomplish this. The third-party SNMP agent can be started before or after starting **naaagt**.

Parameters

-E *priority* Sets the registration priority. If two subagents register the same OIDs, requests are sent to the subagent with the highest priority or to the subagent with the same priority that was the last to register. The *priority* range is zero (high) to 255 (low). Default: zero.

-K Causes the Native Agent Adapter (**naaagt**) to continue to run, even if the HP SNMP Master Agent (**snmpdm**) terminates. When **snmpdm** subsequently restarts, **naaagt** will re-establish communication with **snmpdm** and re-register its OIDs. This option is recommended if the **Independent** Start-up procedure is used, described below. By default, **naaagt** terminates automatically when **snmpdm** terminates.

Troubleshooting Parameters

-h
-help Displays usage information.

-m *logMask* Sets the logging mask. **logMask** may be one of the following names: **APWARN**, **APERROR**, **APTRACE**, **APALL**, or **APNONE**. Multiple logging options can be specified by repeating the **-m** option. Logging data is written to the SNMP log file, **/var/adm/snmpd.log**. Default: **APNONE**.

-n Causes **naaagt** to run in foreground. By default, **naaagt** runs in the background.

Native Agent Adapter Start-up

The Native Agent Adapter start-up procedure may be provided by the vendor of the third-party SNMP agent. Refer to third-party documentation for instructions.

Automatic Start-up

The third-party SNMP agent and its Native Agent Adapter can be started and stopped automatically during system start-up and shutdown. This is accomplished by providing a start-up/shutdown script under the **/sbin/init.d** directory, with symbolic links to it under an appropriate **/sbin/rc***N***.d** directory. See the *rc*(1M) man page for details. See the *Independent*Start-up section for procedures for starting **naaagt**.

Manual Start-up

The third-party SNMP agent and its Native Agent Adapter can be started using the **/usr/sbin/snmpd** command by providing start-up and shutdown scripts under **/sbin/SnmpAgtStart.d**. Normally,

n

those scripts are merely symbolic links to the automatic start-up/shutdown script under **/sbin/init.d**, following the same naming conventions documented in the *rc*(1M) manual page. These scripts are executed by the **/usr/sbin/snmpd** command. See the *Independent*Start-up section for procedures for starting **naaagt**.

Independent Start-up

The third-party SNMP agent and its Native Agent Adapter can be started by entering commands directly or by executing an arbitrary script. When this approach is used, the **naaagt**-K option should probably be used because the third-party agent and adapter will not be restarted by the **/usr/sbin/snmpd** command.

The following procedure should be used to start **naaagt** for each third-party agent.

- Export **HP_NAA_PORT**, which must be set to a unique port number.
- Export **HP_NAA_CNF**, which must be set to an absolute path name for a third-party-specific **naaCnf** file. Create the **naaCnf** file; refer to third-party documentation for a list of the OIDs that are instrumented by the third-party SNMP agent.
- Export **HP_NAA_GET_COMMUNITY**, which must be set to the community name to be used in SNMP requests forwarded from **naaagt** to the third-party SNMP agent. This environment variable is required only if the third-party agent is configured to use a community name other than the default ("public").
- Create a unique symbolic link to **/usr/sbin/naaagt**. This makes it convenient to distinguish each Native Agent Adapter in output from the **ps -ef** command. Execute the symbolic link to start **naaagt**.
- Start the third-party SNMP agent, following procedures in the third-party documentation. This can be done before or after starting **naaagt**.

n

naaCnf File Format

The **naaCnf** file consists of a list of numeric object identifiers (OIDs), one OID per line. Each OID is a subtree of MIB variables that are instrumented by the third-party SNMP agent. Refer to third-party documentation for the list of OIDs. Blank lines and lines beginning with "#" are treated as comments. Leading and trailing spaces on a line are ignored. The OID can start with an optional period.

Example

```
# RDBMS MIB (with leading period)
.1.3.6.1.2.1.39
# Third-party Private MIB (without leading period)
1.3.6.1.4.1.111
# application/applTable MIB
1.3.6.1.2.1.27.1.1
```

EXTERNAL INFLUENCES

Environment Variables

HP_NAA_CNF The absolute path name for the **naaCnf** file from which the Native Agent Adapter reads the OIDs to be registered for its third-party SNMP agent. Default (not recommended): **/etc/SnmpAgent.d/naa.cnf**.

HP_NAA_GET_COMMUNITY
The community name that the Native Agent Adapter uses in SNMP requests forwarded to the third-party SNMP agent. Note that this does not have to match any community names accepted by the SNMP Master Agent, which are defined in **/etc/SnmpAgent.d/snmpd.conf**. This environment variable is required only if the standard community name ("public") is not accepted by the third-party SNMP agent. Refer to third-party documentation for instructions regarding SNMP community names. Default: public.

HP_NAA_PORT The non-standard UDP port number to which the Native Agent Adapter forwards SNMP requests to the third-party SNMP agent. This must match the port number that the third-party agent listens to instead of the standard SNMP port (161). Each third-party SNMP agent must listen to a different non-standard UDP port number. Default: 8161.

International Code Set Support

Supports single-byte character code sets except where the SNMP protocol supports only 7-bit characters encoded in ASCII.

WARNINGS

The Native Agent Adapter only supports SNMP read requests (for example, SNMP Get). SNMP Set requests must be sent directly to the third-party SNMP agent's non-standard UDP port (**HP_NAA_PORT**).

If the **HP_NAA_PORT** value is not a valid port number, **naaagt** terminates without writing any error message either to the display or to **/var/adm/snmpd.log**.

If there are no valid OIDs in the **naaCnf** file, **naaagt** terminates. It does not register a default OID.

There may be unexpected results if the **-E** *priority* is outside the valid range.

AUTHOR

naaagt was developed by SNMP Research and Hewlett-Packard Company.

FILES

/etc/SnmpAgent.d/naa.cnf	Default **naaCnf** file. Not recommended.
/sbin/SnmpAgtStart.d	Directory for SNMP start-up and shutdown scripts.
/usr/sbin/naaagt	Native Agent Adapter command.
/usr/sbin/snmpdm	HP SNMP Master Agent command.
/var/adm/snmpd.log	HP SNMP log file.

SEE ALSO

rc(1M), snmpd(1M), snmpd.conf(4).

NAME

named-xfer - ancillary agent for inbound zone transfers

SYNOPSIS

named-xfer -z *zone_to_transfer* **-f** *db_file* **-s** *serial_no* [**-d** *debuglevel*] [**-l** *debug_log_file*] [**-t** *trace_file*] [**-p** *port#*] [**-S**] *nameserver*...

DESCRIPTION

named-xfer is an ancillary program executed by *named*(1M) to perform an inbound zone transfer. It is rarely executed directly, and then generally by system administrators trying to debug a zone transfer problem. See RFC's 1033, 1034, and 1035 for more information on the Internet name-domain system.

Options are:

-z specifies the name of the zone to be transferred.

-f specifies the name of the file into which the zone should be dumped when it is received from the primary server.

-s specifies the serial number of the current copy of the zone selected for transfer. If the SOA resource record received from the specified remote nameserver(s) does not have a serial number higher than one specified with this option, the transfer will be aborted.

-d Print debugging information. A number after the **d** determines the level of messages printed.

-l Specifies a log file for debugging messages. The default file uses the prefix **xfer.ddt.** and is located in the **/var/tmp** directory. Note this option only applies if **-d** is also specified.

-t Specifies a trace file which will contain a protocol trace of the zone transfer. This is probably only of interest when debugging the name server itself.

-p Use a different port number. The default is the standard port number as returned by *getservbyname*(3) for service "domain".

-S Perform a restricted transfer of only the SOA, NS and glue A records for the zone. The SOA will not be loaded by **named**, but will be used to determine when to verify the NS records. See the **stubs** directive in *named*(1M) for more information.

Additional arguments are taken as name server addresses in so-called "dotted-quad" syntax only; no host name are allowed here. At least one address must be specified. Any additional addresses will be tried in order if the first one fails to transfer to us successfully.

RETURN VALUE

0 Indicates that the zone was up-to-date and no transfer was needed.

1 Indicates a successful transfer.

2 Indicates that the host(s) **named-xfer** queried cannot be reached or that an error occurred and **named-xfer** did not log a corresponding error message.

3 Indicates that an error occurred and **named-xfer** logged an error message.

AUTHOR

named-xfer was developed by the University of California, Berkeley.

SEE ALSO

named(1M), resolver(3N), resolver(4), hostname(5).

RFC 882, RFC 883, RFC 973, RFC 974, RFC 1033, RFC 1034, RFC 1035, RFC 1123.

NAME

named - Internet domain name server

SYNOPSIS

named [**-d** *debuglevel*] [**-p** *port_number*] [[**-**(**b|c**)] *config_file*] [**-f**] [**-q**] [**-r**]
[**-u** *user_name*] [**-g** *group_name*] [**-t** *directory*] [**-w** *directory*] [*config_file*]

DESCRIPTION

named is the Internet domain name server. See RFC1033, RFC1034 and RFC1035 for more information on the Domain Name System. Without any arguments, **named** reads the default configuration file **/etc/named.conf**, reads any initial data, and listens for queries.

Options are:

-d *debuglevel*
: Print debugging information. A number after the **d** determines the level of messages printed. If negative, *debuglevel* is set to “1”.

  NOTE: The new debugging framework is considerably more sophisticated than it was in older versions of **NAMED**. The configuration file's **logging** statement allows for multiple, distinct levels of debugging for each of a large set of categories of events (such as queries, transfers in or out, etc.).

-p *port_number*
: Use a different port number.

  NOTE: Previously, the syntax: " **-p** *port#*[*/localport#*] " was supported; the first port was that used when contacting remote servers, and the second one was the service port bound by the local instance of **NAMED**. The current usage is equivalent to the old usage without the *localport#* specified; this functionality can be specified with the **listen-on** clause of the configuration file's **options** statement.

-(**b** | **c**) *config_file*
: Use a *config_file* other than **/etc/named.conf**.

-f
: Run this process in the foreground; don't *fork*(2) and daemonize. (The default is to daemonize.)

-q
: Trace all incoming queries, if **NAMED** has been compiled with **QRYLOG** defined.

  NOTE: This option is deprecated in favor of the **queries** logging category of the configuration file's **logging** statement.

-r
: Turns recursion off in the server. Answers can only come from local (primary or secondary) zones. This can be used on root servers. The default is to use recursion.

  NOTE: This option can be overridden by and is deprecated in favor of the **recursion** clause of the configuration file's **options** statement.

-u *user_name*
: Specifies the user the server should run as after it initializes. The value specified may be either a username or a numeric user id. If the **-g** flag is not specified, then the group id used will be the primary group of the user specified (**initgroups()** is called, so all of the user's groups will be available to the server).

-g *group_name*
: Specifies the group the server should run as after it initializes. The value specified may be either a groupname or a numeric group id.

-t *directory*
: Specifies the directory the server should **chroot()** into as soon as it is finished processing command line arguments.

-w *directory*
: Sets the working directory of the server. The **directory** clause of the configuration file's **options** statement overrides any value specified on the command line. The default working directory is the current directory (**.**).

Any additional argument is taken as the name of the configuration file. The configuration file contains information about where the name server gets its initial data. If multiple configuration files are specified, only the last is used. Lines in the configuration file cannot be continued on subsequent lines. The following is a small example:

```
//
/*     configuration file for name server     */
#
options {
        directory "/usr/local/domain";
        forwarders {
                10.0.0.78;
                10.2.0.78;
        };
        noforward {
                test.com;
                c.b.a.in-addr.arpa;
        };
        sortlist {
                10.0.0.0;
                26.0.0.0;
        };
};

zone "." {
        type hint;
        file "db.cache";
};

zone "berkeley.edu" {
        type master;
        file "db.berkeley";
};

zone "32.128.in-addr.arpa" {
        type master;
        file "db.128.32";
};

zone "cc.berkeley.edu" {
        type slave;
        file "db.cc";
        masters {
                128.32.137.8;
        };
};

};
```

The **directory** statement causes the server to change its working directory to the directory specified. This can be important for the correct processing of **$INCLUDE** files (described later) in primary server's master files.

Files referenced in the configuration file contain data in the master file format described in RFC1035.

The **forwarders** line specifies the addresses of sitewide servers that will accept recursive queries from other servers. If the configuration file specifies one or more forwarders, then the server will send all queries for data not in the cache or in its authoritative data to the forwarders first. Each forwarder will be asked in turn until an answer is returned or the list is exhausted. If no answer is forthcoming from a forwarder, the server will continue as it would have without the forwarders line unless it is in **forward-only** mode. The forwarding facility is useful to cause a large sitewide cache to be generated on a master, and to reduce traffic over links to outside servers.

The **noforward** line specifies that the DNS server will not forward any request for something in or below the listed domains, even if the forwarders directive exists.

The **sortlist** line can be used to indicate networks that are preferred over other, unlisted networks. Address sorting only happens when the query is from a host on the same network as the server. The best address is placed first in the response. The address preference is local network addresses, then addresses on the sort list, then other addresses.

A server can access information from servers in other domains given a list of root name servers and their addresses. The **zone "."** line specifies that data in **db.cache** is to be placed in the backup cache. Its use is to prime the server with the locations of root domain servers. This information is used to find the current root servers and their addresses. The current root server information is placed in the operating cache. Data for the root nameservers in the backup cache are never discarded.

The **zone "berkeley.edu"** line states that the master file **db.berkeley** contains authoritative data for the **berkeley.edu** zone. A server authoritative for a zone has the most accurate information for the zone. All domain names are relative to the origin, in this case, **berkeley.edu** (see below for a more detailed description).

The **zone "32.128.in-addr.arpa"** line states that the file **db.128.32** contains authoritative data for the domain **32.128.in-addr.arpa**. This domain is used to translate addresses in network **128.32** to hostnames.

The **zone "cc.berkeley.edu"** line specifies that all authoritative data in the **cc.berkeley.edu** zone is to be transferred from the name server at Internet address 128.32.137.8 and will be saved in the backup file **db.cc**. Up to 10 addresses can be listed on this line. If a transfer fails, it will try the next address in the list. The secondary copy is also authoritative for the specified domain. The first non-Internet address on this line will be taken as a filename in which to backup the transferred zone. The name server will load the zone from this backup file (if it exists) when it boots, providing a complete copy, even if the master servers are unreachable. Whenever a new copy of the domain is received from one of the master servers, this file is updated. If no file name is given, a temporary file will be used and will be deleted after each successful zone transfer. This is not recommended because it causes a needless waste of bandwidth.

Master File Format

The master file consists of control information and a list of resource records for objects in the zone of the forms:

$INCLUDE *filename opt_domain*
$ORIGIN *domain*
domain opt_ttl opt_class type resource_record_data

where:

domain	is **.** for root, **@** for the current origin, or a standard domain name. If domain is a standard domain name that does not end with ".", the current origin is appended to the domain. Domain names ending with "." are unmodified.
opt_domain	This field is used to define an origin for the data in an included file. It is equivalent to placing an **$ORIGIN** statement before the first line of the included file. The field is optional. Neither the opt_domain field nor **$ORIGIN** statements in the included file modify the current origin for this file.
opt_ttl	An optional integer number for the time-to-live field. It defaults to zero, meaning the minimum value specified in the SOA record for the zone.
opt_class	The object address type; currently only one type is supported, IN, for objects connected to the DARPA Internet.
type	This field contains one of the following tokens; the data expected in the *resource_record_data* field is in parentheses:

A	a host address (dotted-quad IP address)
NS	an authoritative name server (domain)
MX	a mail exchanger (domain), preceded by a preference value (0..32767), with lower numeric values representing higher logical preferences.
CNAME	the canonical name for an alias (domain)
SOA	marks the start of a zone of authority (domain of originating host, domain address of maintainer, a serial number and the following parameters in seconds: refresh, retry, expire and minimum TTL (see RFC 883)).

n

NULL	a null resource record (no format or data)
RP	a Responsible Person for some domain name (mailbox, TXT-referral)
PTR	a domain name pointer (domain)
HINFO	host information (cpu_type OS_type)
TXT	text data (string)
WKS	a well known service description (IP address followed by a list of services)

Resource records normally end at the end of a line, but may be continued across lines between opening and closing parentheses. Comments are introduced by semicolons and continue to the end of the line.

NOTE: There are other resource record types not shown here. You should consult the BIND Operations Guide ("BOG") for the complete list. Some resource record types may have been standardized in newer RFC's but not yet implemented in this version of BIND.

SOA Record Format

Each master zone file should begin with an SOA record for the zone. An example SOA record is as follows:

```
@   IN      SOA     ucbvax.Berkeley.EDU.  rwh.ucbvax.Berkeley.EDU. (
                                  1989020501      ; serial
                                  10800   ; refresh
                                  3600    ; retry
                                  3600000 ; expire
                                  86400 ) ; minimum
```

The SOA specifies a serial number, which should be changed each time the master file is changed. Note that the serial number can be given as a dotted number, but this is a very unwise thing to do since the translation to normal integers is via concatenation rather than multiplication and addition. You can spell out the year, month, day of month, and 0..99 version number and still fit inside the unsigned 32-bit size of this field. (It's true that we will have to rethink this strategy in the year 4294.)

Secondary servers check the serial number at intervals specified by the refresh time in seconds; if the serial number changes, a zone transfer will be done to load the new data. If a master server cannot be contacted when a refresh is due, the retry time specifies the interval at which refreshes should be attempted. If a master server cannot be contacted within the interval given by the expire time, all data from the zone is discarded by secondary servers. The minimum value is the time-to-live ("TTL") used by records in the file with no explicit time-to-live value.

NOTE: The boot file directives **domain** and **suffixes** have been obsoleted by a more useful, resolver-based implementation of suffixing for partially-qualified domain names. The prior mechanisms could fail under a number of situations, especially when then local nameserver did not have complete information.

The following signals have the specified effect when sent to the server process using the *kill*(1) command:

SIGHUP	Causes server to read **named.conf** and reload database. If the server is built with the **FORCED_RELOAD** compile-time option, then **SIGHUP** will also cause the server to check the serial number on all secondary zones; normally, the serial numbers are only checked at the SOA-specified intervals.
SIGINT	Dumps current data base and cache to **/var/tmp/named_dump.db**.
SIGILL	Dumps statistics data into **named.stats** if the server is compiled with **-DSTATS**. Statistics data is appended to the file.
SIGSYS	Dumps the profiling data in **/var/tmp** if the server is compiled with profiling (server forks, chdirs and exits).
SIGTERM	Dumps the primary and secondary database files. Used to save modified data on shutdown if the server is compiled with dynamic updating enabled.
SIGUSR1	Turns on debugging; each **SIGUSR1** increments debug level.
SIGUSR2	Turns off debugging completely.
SIGWINCH	Toggles the logging of all incoming queries via **syslog()** (requires server to have been built with the **QRYLOG** option).

sig_named(1M) can also be used for sending signals to the server process.

NOTE

named checks for symbolic or hard links while writing to the following files:

```
/var/run/named.pid
/var/tmp/named_dump.db
/var/tmp/named.run
/var/tmp/named.stats
```

If any of these files is linked to a different file before **named** writes into them, **named** will delete that file and create it afresh.

DIAGNOSTICS

Any errors encountered by **named** in the configuration file, master files, or in normal operation are logged with **syslog** and in the debug file, **/var/tmp/named.run**, if debugging is on.

AUTHOR

named was developed by the University of California, Berkeley.

FILES

/etc/named.conf	name server configuration file
/var/run/named.pid	process ID
/var/tmp/named.run	debug output
/var/tmp/named_dump.db	dump of the name server database
/var/tmp/named.stats	nameserver statistics data

SEE ALSO

kill(1), hosts_to_named(1M), named-xfer(1M), sig_named(1M), signal(2), gethostent(3N), resolver(3N), resolver(4), hostname(5),

RFC 882, RFC 883, RFC 973, RFC 974, RFC 1032, RFC 1033, RFC 1034, RFC 1035, RFC 1123.

NAME

ncheck - generate a list of path names from inode numbers

SYNOPSIS

/usr/sbin/ncheck [**-F** *FStype*] [**-V**] [**-o** *specific_options*] [*special* ...]

DESCRIPTION

ncheck, when invoked without arguments, generates a list of path names corresponding to the inode numbers of all files contained on the file systems listed in **/etc/fstab.** If *special* is specified, ncheck reports on the *special* only. Path names generated by **ncheck** are relative to the given *special*.

Options

-F *FStype* Specify the file system type on which to operate (see *fstyp*(1M) and *fs_wrapper*(5)). If this option is not included on the command line, then the file system type is determined from the file **/etc/fstab** by matching each *special* with an entry in that file. If there is no entry in **/etc/fstab**, then the file system type is determined from the file **/etc/default/fs**.

-o *specific_options*
Specify options specific to each file system type. *specific_options* is a list of suboptions and/or keyword/attribute pairs intended for a specific *FStype*-specific module of the command. See the file-system-specific manual pages for a description of the *specific_options* supported, if any.

-V Echo the completed command line, but perform no other action. The command line is generated by incorporating the user-specified options and other information derived from **/etc/fstab**. This option allows the user to verify the command line.

EXAMPLES

Execute the **ncheck** command on all *special* in **/etc/fstab:**

```
ncheck
```

Execute the **ncheck** command on HFS file system **/dev/dsk/c1d2s0:**

```
ncheck -F hfs /dev/dsk/c1d2s0
```

Display a completed command line without executing the command:

```
ncheck -V /dev/dsk/c1d2s0
```

FILES

/etc/default/fs	Specifies the default system type.
/etc/fstab	Static information about the file systems.

AUTHOR

ncheck was developed by AT&T and HP.

SEE ALSO

fstab(4), fstyp(1M), fs_wrapper(5), ncheck_hfs(1M), ncheck_vxfs(1M).

STANDARDS CONFORMANCE

ncheck: SVID2, SVID3

NAME

ncheck - generate a list of path names from inode numbers for a HFS file system

SYNOPSIS

/usr/sbin/ncheck [**-F hfs**] [**-V**] [**-S sector_ranges**] [**-i** *inode-numbers*]
[**-a**] [**-s**] [*special* ...]

DESCRIPTION

ncheck, when invoked without arguments, generates a list of path names corresponding to the inode numbers of all files contained on the HFS file systems listed in **/etc/fstab**. If *special* is specified, ncheck reports on the *special* only. Path names generated by **ncheck** are relative to the given *special*. Names of directory files are followed by **/**.

Options

-a Allow printing of the names **.** and **..**, which are ordinarily suppressed.

-F hfs Specify the HFS file system type.

-i *inode-numbers*
Report only on files whose inode numbers are specified on the command line, in *inode-numbers*. *inode-numbers* is a comma separated list of inode numbers.

-s Report only on special files and regular files with set-user-ID mode. The **-s** option is intended to discover concealed violations of security policy.

-V Echo the completed command line, but performs no other action. The command line is generated by incorporating the user-specified options and other information derived from **/etc/fstab**. This option allows the user to verify the command line.

-S *sector_ranges*
Report only on files using sector numbers specified on the command line in *sector_ranges*. *sector_ranges* is a comma separated list of sector ranges. A sector range is a starting sector number and an ending sector number separated by a dash, or just a sector number. The sector numbers should be in DEV_BSIZE units. If no pathname contains the sector number it will be reported as free or containing file system structure. Sectors beyond the end of the file system will be reported as illegal.

Access Control Lists

Continuation inodes (that is, inodes containing additional access control list information) are quietly skipped since they do not correspond to any path name.

EXAMPLES

Execute the **ncheck** command on all *special* in **/etc/fstab:**

```
ncheck
```

Execute the **ncheck** command on HFS file system **/dev/dsk/c1d2s0**:

```
ncheck -F hfs /dev/dsk/c1d2s0
```

EXTERNAL INFLUENCES

International Code Set Support

Single- and multi-byte character code sets are supported.

DIAGNOSTICS

When the file system structure is improper, **??** denotes the “parent” of a parentless file and a path-name beginning with **...** denotes a loop.

AUTHOR

ncheck was developed by AT&T and HP.

FILES

/etc/default/fs	Specifies the default file system type.
/etc/fstab	Static information about the file systems.

SEE ALSO

acl(5), fsck(1M), fstab(4), fs_wrapper(5), ncheck(1M), sort(1).

STANDARDS CONFORMANCE

ncheck: SVID2, SVID3

NAME

ncheck - generate pathnames from inode numbers for a VxFS file system

SYNOPSIS

/usr/sbin/ncheck [**-a**] [**-F vxfs**] [**-i** *ilist*] [**-o** *specific_options*] [**-s**] [**-S** *sector_list*] [**-V**] *special* ...

DESCRIPTION

ncheck generates a list of pathnames corresponding to inode numbers for files in a specified VxFS file system.

You can specify a *range* for some options. A range can be a single number, or two numbers separated by a hyphen (**-**). The range is inclusive. If the range is a single number, the output will refer to the single sector, block, or surface specified by that number. If you enter a hyphen and omit the first number (**-***number*), the range begins at zero. If you omit the second number, the range ends at the end of the file system.

Names of directory files are followed by slash dot (**/.**).

Options

-a Allow printing of the names dot (**.**) and dot dot (**..**), which are ordinarily suppressed.

-F vxfs Specify the file-system type (**vxfs**).

-i *ilist* Limit the report to the files on the inode list, *ilist*, that follows. The *ilist* must be separated by commas without spaces.

-o *specific_options*
Specify options specific to the VxFS file-system type. *specific_options* is a list of suboptions and/or keyword/attribute pairs intended for the VxFS-specific module of the command.

The available options are

m Print mode information (used in conjunction with **-i** option).

b=*block*
Print the pathname containing file system block number *block*.

block=*block_range*
Print information on all inodes containing or referencing block numbers in the specified range. The output format is the same as that for **-o sector=**, but the units used are file-system blocks instead of sectors.

sector=*sector_range*
Report on all inodes containing or referencing the sector(s) in *sector_range*. The output includes the inode number, fileset index of the inode, sector(s) contained and the pathname or inode type. Inodes searched include structural inodes and attribute inodes, so a pathname is only generated when the sector is contained by a file. If the sector is not contained in any file, the inode type is printed as **<free>**. Multiple **-o sector=** options accumulate.

surface[=*sector_range***]**
Perform a surface analysis. If a *sector_range* is specified, perform a surface analysis only for that range. All the sectors are read and if the read of a sector fails, its sector number is printed. If any bad sectors are found, **ncheck** treats the list of bad sectors as input to the **-o sector=***sector_range* option and produces a list of inodes containing or referencing the bad sectors.

-s Report only on special files and regular files with set-user-ID mode. This option may be used to detect violations of security policy.

-S *sector_list* Report on files containing or referencing the specified sector(s). Output consists of the fileset name, fileset index, inode number, and pathname of file or file type if a structural inode or attribute inode. Sectors not allocated to any file or file system structure are reported as **<free>**. Sectors not part of the file system are reported as **<unused>**. Unused or irrelevant fields are printed as **-**.

sector_list consists of one or more *ranges* of sector numbers, separated by commas without intervening spaces. Multiple **-S** options accumulate.

-V Echo the completed command line, but do not execute the command. The command line is generated by incorporating the user-specified options and other information derived from **/etc/fstab**. This option allows the user to verify the command line.

EXAMPLES

Report on all inodes or file system structures containing or referencing sector 20 through 35 (inclusive) in the file system **/dev/vg01/rlvol1**:

```
ncheck -F vxfs -osector=20-35 /dev/vg01/rlvol1
```

Same as above but report on all inodes or file system structures referencing any sector in the file system **/dev/vg01/rlvol1**:

```
ncheck -F vxfs -osector= /dev/vg01/rlvol1
```

Report on a specified range of block numbers (partial output shown):

```
ncheck -F vxfs -oblock=- /dev/vg00/lvol6

/dev/vg00/lvol6:

sectors(348160)          blocks(348160)
-----------------        -----------------
0-348159                 0-348159

fileset    fset        mtch  match
name       indx  inode indx  inode blocks        name
---------- ---- ------ ---- ------ ------------- -----------------
STRUCTURAL    1      3    -     35 9-10          <fileset_header>
STRUCTURAL    1      4    1      - 11-14         <inode_alloc_unit>
STRUCTURAL    1      5    1     37 1296-1303     <inode_list>
STRUCTURAL    1      5    1     37 24-31         <inode_list>
STRUCTURAL    1      5    1     37 1288-1295     <inode_list>
STRUCTURAL    1      5    1     37 16-23         <inode_list>
STRUCTURAL    1      6    -      - 15            <current_usage_tbl>
STRUCTURAL    1      7    -     39 33            <object_loc_tbl>
STRUCTURAL    1      8    -     40 34            <device_config>
STRUCTURAL    1      9    -     41 40-1063       <intent_log>
STRUCTURAL    1     10    -     42 1144-1151     <extent_map>
STRUCTURAL    1     10    -     42 1072-1143     <extent_map>
STRUCTURAL    1     10    -     42 1064-1071     <extent_map>
STRUCTURAL    1     32    -      - 32            <state_alloc_bitmap>
STRUCTURAL    1     33    -      - 1284          <device_label>
STRUCTURAL    1     33    -      - 0-8           <device_label>
STRUCTURAL    1     34    -      - 35            <extent_au_summary>
STRUCTURAL    1     35    -      3 1280-1281     <fileset_header>
STRUCTURAL    1     37    1      5 1296-1303     <inode_list>
STRUCTURAL    1     37    1      5 24-31         <inode_list>
STRUCTURAL    1     37    1      5 1288-1295     <inode_list>
STRUCTURAL    1     37    1      5 16-23         <inode_list>
STRUCTURAL    1     39    -      7 1282          <object_loc_tbl>
STRUCTURAL    1     40    -      8 1283          <device_config>
STRUCTURAL    1     41    -      9 40-1063       <intent_log>
STRUCTURAL    1     42    -     10 1144-1151     <extent_map>
STRUCTURAL    1     42    -     10 1072-1143     <extent_map>
STRUCTURAL    1     42    -     10 1064-1071     <extent_map>
STRUCTURAL    1     64  999      - 36-39         <inode_alloc_unit>
STRUCTURAL    1     65  999     97 59072-59135   <inode_list>
STRUCTURAL    1     65  999     97 1152-1159     <inode_list>
STRUCTURAL    1     69  999      - 1160-1167     <bsd_quota>
STRUCTURAL    1     97  999     65 59072-59135   <inode_list>
STRUCTURAL    1     97  999     65 1152-1159     <inode_list>
UNNAMED     999     96    -      - 4861          /file1
UNNAMED     999   1807    -      - 344387        /file1000
UNNAMED     999   1822    -      - 328082        /file2
-             -      -    -      - 19766-19769   <free>
-             -      -    -      - 347192-347194 <free>
```

DIAGNOSTICS

When the file system structure is not correct, **ncheck** prints **???** to denote the "parent" of a parentless file. A pathname beginning with **...** denotes a loop. A pathname beginning with ******* denotes a directory entry whose **..** (dotdot) entry does not correspond with the directory in which it was found.

FILES

/etc/fstab Static information about the file systems.

SEE ALSO

sort(1), fsck(1M), fsck_vxfs(1M), ncheck(1M).

NAME

ndd - network tuning

SYNOPSIS

ndd -get *network_device parameter*

ndd -set *network_device parameter value*

ndd -h sup[**ported**]

ndd -h unsup[**ported**]

ndd -h [*parameter*]

ndd -c

DESCRIPTION

The **ndd** command allows the examination and modification of several tunable parameters that affect networking operation and behavior. It accepts arguments on the command line or may be run interactively. The **-h** option displays all the supported and unsupported tunable parameters that **ndd** provides. Valid *network_device* names are: **/dev/arp**, **/dev/ip**, **/dev/rawip**, **/dev/tcp**, and **/dev/udp**. Set *parameter* to **?** to get a list of parameters for a particular *network_device*.

ndd -get Get the value of the *parameter* for *network_device* and print the *value* to standard output. Returned numbers are always displayed as decimal strings.

ndd -set Set *parameter* for *network_device* to *value*.

All times are specified in milliseconds, e.g. 240000 for 4 minutes. Unless stated otherwise, numbers are assumed to be in decimal. Use "0x" prefix to specify hexadecimal values.

In general, all tunable parameters are global, i.e., they affect all instances of the network module. Some settings take effect immediately, while others are used to initialize data for an instance and will only affect newly opened streams.

ndd -h supported
Display all the supported tunable parameters. This set of parameters are supported by HP and detailed descriptions of these tunable parameters are available through the **-h** *parameter* command.

ndd -h unsupported
Display all the unsupported tunable parameters. This set of parameters are not supported by HP and modification of these tunable parameters are not suggested nor recommended. Setting any unsupported tunable parameters on your system may result in adverse effects to your networking operations.

ndd -h When *parameter* is specified, a detail description of the *parameter*, along with its minimum, maximum, and default value are displayed. If no parameter is specified, it displays all supported and unsupported tunable parameters.

ndd -c Read input from the configuration file **/etc/rc.config.d/nddconf** and set the tunable parameters. A user may specify tunable parameters in the nddconf configuration file, and these parameters will be set automatically each time the system boots.

DIAGNOSTICS

When the command fails, an error message is printed to the standard error and the command terminates with an exit value of one.

WARNINGS

Care must be used when setting parameters for a network_device. Setting a tunable parameter to an inappropriate value can result in adverse affects to your networking operations.

EXAMPLES

To get help information on all supported tunable parameters:

```
ndd -h supported
```

To get a detail description of the tunable parameter, **ip_forwarding**:

```
ndd -h ip_forwarding
```

To get a list of all TCP related parameters:

```
ndd -get /dev/tcp ?
```

To get the current value of the tunable parameter, **ip_forwarding**:

```
ndd -get /dev/ip ip_forwarding
```

To set the value of the default TTL parameter for UDP to 128:

```
ndd -set /dev/udp udp_def_ttl 128
```

FILES

/etc/rc.config.d/nddconf	Contains tunable parameters that will be set automatically each time the system boots.

AUTHOR

ndd was developed by HP.

NAME

netfmt - format tracing and logging binary files

SYNOPSIS

/usr/sbin/netfmt [**-k**] **-s** [**-t** *records*] [[**-f**] *file_name*]

/usr/sbin/netfmt [**-k**] **-p** [**-c** *config_file*]

/usr/sbin/netfmt [**-c** *config_file*] [**-F**] [**-t** *records*] [**-v**] [**-l**] [**-n**]
[**-N** | [**-1** [**-L**] [**-T**]]] [[**-f**] *file_name*]

/usr/sbin/netfmt -k [**-c** *config_file*] [**-F**] [**-t** *records*] [**-v**] [[**-f**] *file_name*]

DESCRIPTION

netfmt is used to format binary trace and log data gathered from the network tracing and logging facility (see *nettl*(1M)) and the kernel logging facility (see *kl*(1M)). The binary trace and log information can be read from a file or from standard input (if standard input is a tty device, an informative message is given and **netfmt** quits). Formatted data is written to standard output.

Formatting options are specified in an optional filter configuration file. Message inclusion and format can be controlled by the filter configuration file. If no configuration commands are specified, all messages are fully formatted.

There are two types of global formatting done by **netfmt**. The first one is global filtering for *NetTL's* trace/log packets and the other is for *KL's* log packets. A description of the filter configuration file follows the option descriptions.

Options

netfmt recognizes the following command-line options and arguments:

-k This option tells **netfmt** that the input file is a *KL* log file. This option should be specified if the user needs to log messages got from *KL* subsystems. This option cannot be specified anywhere except as the first option in the command line.

-s Display a summary of the input file. The summary includes the total number of messages, the starting and ending timestamps, the types of messages, and information about the system that the data was collected on. The contents of the input file are not formatted; only a summary is reported.

-t *records* Specifies the number of records from the tail end of the input file to format. This allows the user to bypass extraneous information at the beginning of the file, and get to the most recent information quickly. The maximum number of *records* that can be specified is 1000. If omitted, all records are formatted. The **-t** option is not allowed when the input file is a FIFO (pipe).

-f *file_name* Specifies the input file containing the binary log or trace data. *file_name* may not be the name of a tty device. Other options may impose additional restrictions on the type of the input file allowed. If omitted, data is read from standard input.

-p Parse input: this switch allows the user to perform a syntax check on the *config_file* specified by the **-c** parameter. All other parameters are ignored. If the syntax is correct, **netfmt** terminates with no output or warnings.

-c *config_file* Specifies the file containing formatter filter configuration commands. Syntax for the commands is given below. When **-c** is omitted the file **$HOME/.netfmtrc** is read for both logging and tracing filter configuration commands if it exists.

-F Follow the input file. Instead of closing the input file when end of file is encountered, **netfmt** keeps it open and continues to read from it as new data arrives. This is especially useful for watching events occur in real time while troubleshooting a problem. Another use would be for recording events to a console or hard-copy device for auditing. (Note that console logging is controlled by the configuration files **/etc/nettlgen.conf** and **/var/adm/conslog.opts**; see *nettlgen.conf*(4).) The **-F** option is not allowed when the input file is redirected.

The following options are not supported by all subsystems. If a subsystem does not support an option, that option is ignored during formatting of data from that subsystem. Consult the product documentation of the subsystem for information regarding the support of these options.

-v	Enables output of verbose information. This includes additional cause and action text with formatted output. This information describes the possible cause of the message and any actions that may be required by the subsystem. After the contents of the input file have been formatted a summary of the file is displayed. When this option is used with the **-t** option, only a summary of the last *records* is reported. No summary is produced when this option is used in conjunction with the **-F** option or if formatting is interrupted.
-l	(*ell*) Turn off inverse video highlighting of certain traced fields. Use this flag when sending formatted trace data to a line printer. By default, certain fields in the trace file are highlighted in inverse video when viewing the formatted trace format at a terminal that supports highlighting.
-n	Shows port numbers and network addresses(such as IP and x121) as numbers (normally, **netfmt** interprets numbers and attempts to display them symbolically).
-N	Enables "nice" formatting where Ethernet/IEEE802.3, SLIP, IP, ICMP, IGMP, TCP, UDP, and RPC packets are displayed symbolically. All remaining user data is formatted in hexadecimal and ASCII.
-1	*(one)* Attempts to tersely format each traced packet on a single line. If **-L** and/or **-T** options are used, the output lines will be more than 80 characters long.
-T	Places a time stamp on terse tracing output. Used with the **-1** (*minus one*) option.
-L	Prefixes local link address information to terse tracing output. Used with the **-1** (*minus one*) option.

Filter Configuration File

Note: Filter configuration file syntax converges the syntax used with the obsolete **nettrfmt** network trace formatter and **netlogfmt** network log formatter commands with new **netfmt** syntax for controlling formatter options. The first section below describes the general use and syntax of the filter configuration file. Specific options for subsystem Naming and Filtering are listed in the *Subsystem Filtering* section below.

The filter configuration file allows specification of two types of information:

- Specify options in order to control how the input data is to be formatted. These options determine what the output looks like and allow a user to select the best format to suit their needs.
- Specify filters in order to precisely tailor what input data is to be discarded and what is to be formatted. **Global filters** control all subsystems; **subsystem filters** pertain only to specific subsystems. There are two types of **Global filters** that **netfmt** supports. The global filtering can start with either the word **formatter**, which means it is global to all the *NetTL's* subsystems and the second type starts with the word **kl_formatter**, which is used to filter *KL's* subsystems.

A filter is compared against values in the input data. If the data matches a filter, the data is formatted; otherwise, the input data is discarded. A filter can also specify *NOT* by using **!** before the filter value in the configuration file. If the input data matches a *NOT* filter, it is discarded. A filter can also be a "wildcard" (matching any value) by specifying an asterisk ***** before the filter value in the configuration file. "Wild card" filters pass all values of the input data. Specifying **!*** as the filter means *NOT ALL*.

Filter Configuration File Syntax

- The formatter ignores white space, such as spaces or tabs. However, newlines (end of line characters) are important, as they terminate comments and filter specifications.
- The formatter is not case sensitive. For example **error** and **ERROR** are treated as equivalent.
- To place comments in the file, begin each comment line with a **#** character. The formatter ignores all remaining characters on that line. There are no inline comments allowed.
- An exclamation point (**!**) in front of an argument indicates *NOT*. This operator is not supported for timestamp, log instance, and ID filtering.
- The asterisk (*****), when used as an argument, indicates *ALL*. Since the default for all formatting options is *ALL*, it is unnecessary to use the asterisk alone. It can be used along with the exclamation point, (**!***) to indicate *NOT ALL*. This operator is not available for timestamp, log instance, and ID filtering.

Global Filtering: For NetTL's Subsystems

The below explained global filtering options apply only to **NetTL's** subsystems. **NetTL's** global filtering commands start with the word **formatter**, followed by the keywords **verbosity**, **mode**, **option**, or **filter**.

formatter verbosity *value*,
value should be either of

high	Enables output of netfmt internal debugging information to standard error. Same as the **-v** option.
low	No internal debugging information is to be displayed.

formatter mode *value*,
value should be one of

raw	Dumps out the messages in hex format.
nice	Enables "nice" formatting. Same as **-N** option.
terse	Attempts to tersely format each traced packet on a single line. Same as **-1** (*minus one*) option.
normal	Normal formatting.

formatter option [**!**] *value*,
value should be

suppress	Normally repeated lines in hex output are condensed into a single line and a message stating that redundant lines have been skipped is displayed. Specifying **!suppress** will print all redundant data. This is useful when the formatted output is used as input into other commands.
highlight	Normally the formatter will highlight certain fields in its trace output in inverse video. Specifying **!highlight** will turn this feature off. Same as the **-l** (*minus ell*) option.

n

formatter filter *type* [**!**] *value* | *****
Six *types* of filtering are provided:

class	log classes
kind	trace kinds
id	connection, process, path, and user
log instance	specific thread of events
subsystem	subsystem names
time	specify ranges of time(s)

The following combinations are recognized:

formatter filter class *value* [*subsystem*]
value indicates the log class. This option allows the user to select one or more classes to be formatted. Initially all log classes are formatted. Only one class is allowed per line. Classes in multiple lines are logically "OR"ed. The optional *subsystem* name sets the class filter only for the specified subsystem. The log classes are:

INFORMATIVE	Describes routine operations and current system values.
WARNING	Indicates abnormal events possibly caused by subsystem problems.
ERROR	Signals an event or condition which was *not* affecting the overall subsystem or network operation, but may have caused an application program to fail.
DISASTER	Signals an event or condition which *did* affect the overall subsystem or network operation, caused several programs to fail or the entire node to shut down.

formatter filter Connection_ID *value*
formatter filter Device_ID *value*
formatter filter Path_ID *value*
formatter filter Process_ID *value*

formatter filter User_ID *value*

value specifies the ID number of the messages to format. Last-entered value has precedence over any previous ones. See the record header in the formatted output to determine which ID numbers to filter on. The **!** operator is *not* allowed in *value*.

formatter filter kind *value* [*subsystem*]

value can either be an established trace kind or a mask. A mask is a hexadecimal representation of a (set of) trace kind(s). Masks in multiple lines are logically "OR"ed. The optional *subsystem* name sets the kind filter only for the specified subsystem. Trace kinds and their corresponding masks are:

Name	Mask	Name	Mask
hdrin	0x80000000	**state**	0x04000000
hdrout	0x40000000	**error**	0x02000000
pduin	0x20000000	**logging**	0x01000000
pduout	0x10000000	**loopback**	0x00800000
proc	0x08000000		

hdrin	Inbound Protocol Header.
hdrout	Outbound Protocol Header.
pduin	Inbound Protocol Data Unit (including header and data).
pduout	Outbound Protocol Data Unit (including header and data).
proc	Procedure entry and exit.
state	Protocol or connection states.
error	Invalid events or condition.
logging	Special kind of trace that contains a log message.
loopback	Packets whose source and destination system is the same.

formatter filter log_instance *value*

value specifies the log instance number of the messages to filter. Selecting a log instance allows the user to see the messages from a single thread of network events. Only one log instance is allowed per filter configuration file. The log instance can not be negated with the **!** operator.

formatter filter subsystem *value*

value specifies the subsystem name. Available subsystem names can be listed by using the command:

```
nettlconf -status
```

Only one subsystem name is allowed per line; multiple lines "OR" the request. To eliminate a given subsystem name, use the **!** operator, which formats all subsystems except those excluded by the list of negated subsystems. To include all subsystems (the default), use the ***** operator. To eliminate all subsystems, use the **!*** operator.

formatter filter time_from *value*
formatter filter time_through *value*

time_from indicates the inclusive starting time. **time_through** indicates the inclusive ending time. *value* consists of *time_of_day* and optionally *day_of_year*, (usually separated by one or more blanks for readability).

time_of_day specifies the time on the 24-hour clock in hours, minutes, seconds and decimal parts of a second (resolution is to the nearest microsecond). Hours, minutes and seconds are required; fractional seconds are optional. *time_of_day* format is *hh*:*mm*:*ss*.*dddddd*.

day_of_year specifies the day of the year in the form month/day/year in the format: *mm*/*dd*/[*yy*]*yy*. Specify month and day numerically, using one or two digits. For example, January can be specified as **1** or **01**; the third day of the month as **3** or **03**. Specify the year in four digits or by its last two digits. Only years in the ranges **1970-2037** are accepted. Two digit years in the range *70-99* are interpreted as being in the *20th century* (19*xx*) and those in the range *00-37* are interpreted as being in the *21st century* (20*xx*) (all ranges inclusive). *day_of_year* is an optional field; the current date is used as a default.

The **time_from** specification includes *only* those records starting from the resolution of time given. For example, if the *time_of_day* for **time_from** is specified as 10:08:00, all times before that, from 10:07:59.999999 and earlier, are excluded from the formatted output. Records with times of 10:08:00.000000 and later are included in the formatted output. Similarly, the **time_through** specification includes *only* up to the resolution of time given. For example, if the *time_of_day* for **time_through** is specified as 10:08:00, all records with times after that, from 10:08:00.000001 onward, are excluded from the formatted output.

Global Filtering: For KL's Subsystems

The below explained global filtering options apply only to *KL's* subystems. *KL's* global filtering commands start with the word **kl_formatter**, followed by either **verbosity**, or **filter**.

kl_formatter verbosity *value*,

value should be either of

high	This will format the packets with the UDD, displayed along with the header of the **KL** packet
low	This will format only the header part of the **KL** packet. No UDD will be formatted. **verbosity** of This will format only the header part of the **KL** packet. No UDD will be formatted. **verbosity** of *low* is default.

kl_formatter filter *type* [**!**] *value* | *****

types of filtering are provided:

class	log classes
processor_id	specific CPU's
process_id	specific process id's
thread_id	specific thread id's
subsystem	subsystem names
time	specify ranges of time(s)

The following combinations are recognized:

kl_formatter filter class *value* [*subsystem*]

value indicates the log class. This option allows the user to select one or more classes to be formatted. Initially all log classes are formatted. Only one class is allowed per line. Classes in multiple lines are logically "OR"ed. The optional *subsystem* name sets the class filter only for the specified subsystem. The log classes are:

INFORMATIVE	Describes routine operations and current system values.
WARNING	Indicates abnormal events possibly caused by subsystem problems.
ERROR	Signals an event or condition which was *not* affecting the overall subsystem or network operation, but may have caused an application program to fail.
DISASTER	Signals an event or condition which *did* affect the overall subsystem or network operation, caused several programs to fail or the entire node to shut down.

kl_formatter filter Processor_ID *value*
kl_formatter filter Process_ID *value*
kl_formatter filter Thread_ID *value*

value specifies the ID number of the messages to format. Last-entered value has precedence over any previous ones. See the record header in the formatted output to determine which ID numbers to filter on. The **!** operator is *not* allowed in *value*.

kl_formatter filter subsystem *value*

value specifies the subsystem name. Available subsystem names can be listed by using the command:

```
nettlconf -status
```

Only one subsystem name is allowed per line; multiple lines "OR" the request. To eliminate a given subsystem name, use the **!** operator, which formats all subsystems except those excluded by the list of negated subsystems. To include all subsystems (the default), use the `*` operator. To eliminate all subsystems, use the `!*` operator.

`kl_formatter filter time_from` *value*
`kl_formatter filter time_through` *value*
The functionality is same as in the case of **`NetTL.`**

Subsystem Filtering

Note: Global filtering described above takes precedence over individual subsystem tracing and logging filtering described below.

Subsystem filters are provided to allow filtering of data for individual subsystems or groups of subsystems. Their behavior varies among individual subsystems. Subsystem filters are valid only when the corresponding subsystems have been installed and configured on the system. See the subsystem documentation for a description of supported subsystem filters and their behavior.

Subsystem filtering commands start with the name of the subsystem followed by the subsystem filter keywords. However, to provide convenience and backwards compatibility, several other filter keywords are provided for the group of LAN subsystems: **`NAME`** and **`FILTER`**. Currently, four types of subsystem filters are provided: LAN, X25, STREAMS, and OTS. The collection of LAN subsystems use the subsystem filters identified by the **`FILTER`** and **`NAME`** keywords and the collection of OTS subsystems use the subsystem filters with the **`OTS`** keyword. The collection of X25 subsystems start their filter commands with the X25 subsystem names.

LAN Naming and Filtering

LAN naming can be used to symbolically represent numbers with more recognizable labels.

`name` *nodename value*
nodename is a character string to be displayed in place of all occurrences of *value*. *value* is a (IEEE802.3/Ethernet) hardware address consisting of 6 bytes specified in hexadecimal (without leading "0x"), optionally separated by **`-`**. **`netfmt`** substitutes all occurrences of *value* with *nodename* in the formatted output. The mapping is disabled when the **`-n`** option is used. This option applies to tracing output only.

LAN filtering is used to selectively format packets from the input file. There are numerous filter types, each associated with a particular protocol layer:

Filter Layer	Filter Type	Description
Layer 1	**`dest`**	hardware destination address
	`source`	hardware source address
	`interface`	software network interface
Layer 2	**`ssap`**	IEEE802.2 source sap
	`dsap`	IEEE802.2 destination sap
	`type`	Ethernet type
Layer 3	**`ip_saddr`**	IP source address
	`ip_daddr`	IP destination address
	`ip_proto`	IP protocol number
Layer 4	**`tcp_sport`**	TCP source port
	`tcp_dport`	TCP destination port
	`udp_sport`	UDP source port
	`udp_dport`	UDP destination port
	`connection`	a level 4 (TCP, UDP) connection
Layer 5	**`rpcprogram`**	RPC program
	`rpcprocedure`	RPC procedure
	`rpcdirection`	RPC call or reply

Filtering occurs at each of the five layers. If a packet matches any filter within a layer, it is passed up to the next layer. The packet must pass every layer to pass through the entire filter. Filtering starts with Layer 1 and ends with Layer 5. If no filter is specified for a particular layer, that layer is "open" and all packets pass through. For a packet to make it through a filter layer which has a filter specified, it must match the filter. Filters at each layer are logically "OR"ed. Filters between layers are logically "AND"ed.

LAN trace and log filters use the following format:

`filter` *type* [**`!`**] *value* | **`*`**
`filter` is the keyword identifying the filter as a LAN subsystem filter.

n

The following filters are available for LAN tracing.

`filter connection` *value*
: *value* takes the form:

  local_addr`:`*port remote_addr*`:`*port*

  where *local_addr* and *remote_addr* can be a hostname or a 4-byte Internet address specified in decimal dot notation (see *inet*(3N) for more information on Internet addresses and decimal dot notations). *port* can be a service name or an *integer*. *integer* represents a port and can be designated by a hexadecimal integer (`0x`*digits*), an octal integer (`0`*digits*), or base-10 integers (0 through 65 535).

`filter dest` *value*
`filter source` *value*
: *value* is a hardware address consisting of 6 bytes specified in hexadecimal (without leading `0x`), optionally separated by `-`.

`filter dsap` *value*
`filter ssap` *value*
: *value* is a hexadecimal integer of the form: `0x`*digit*; an octal integer of the form: `0`*digits*; or a base-ten integer, 0 through 255.

`filter interface` *value*
: *value* identifies a network interface and takes the form: `lan`*n* for LAN interface, or `lo`*n* for loopback interface, where *n* is the logical unit number, as in `lan0`.

`filter ip_daddr` *value*
`filter ip_saddr` *value*
: *value* is a hostname or a 4-byte Internet address specified in decimal dot notation (see *inet*(3N) for more information on Internet addresses and decimal dot notations).

`filter ip_proto` *value*
: *value* is a hexadecimal integer of the form: `0x`*digit*; an octal integer of the form: `0`*digits*; or a base-ten integer, 0 through 255 (see *protocols*(4) for more information on protocol numbers).

n

`filter tcp_dport` *value*
`filter tcp_sport` *value*
`filter udp_dport` *value*
`filter udp_sport` *value*
: *value* is a port number designated as a 2-byte integer value or a service name. The integer value can be designated by a hexadecimal integer (`0x`*digits*), an octal integer (`0`*digits*), or a base-10 integer (0 through 65 535).

`filter rpcprogram` *value*
: *value* is a RPC program name or an integer RPC program number (see *rpc*(4) for more information on RPC program names). The integer value can be designated by a hexadecimal integer (`0x`*digits*), an octal integer (`0`*digits*), or a base-10 integer (0 through 65 535).

`filter rpcprocedure` *value*
: *value* is an integer RPC procedure number. The integer value can be designated by a hexadecimal integer (`0x`*digits*), an octal integer (`0`*digits*), or a base-10 integer (0 through 65 535).

`filter rpcdirection` *value*
: *value* can be either `call` or `reply`.

`filter type` *value*
: *value* is a hexadecimal integer of the form: `0x`*digits*; an octal integer of the form: `0`*digits*; or a base-ten integer (0 through 65 535).

LAN log filtering command has the following form:

`filter subsystem` *value*
: *value* takes the form:

  subsys_name **`event`** *event_list*

  where *subsys_name* is a subsystem name obtained using the **`nettlconf -status`** command or one of the following abbreviations:

axin	**bufs**	**caselib**	**caserouter**
ip	**ipc**	**lan**	**loopback**
netisr	**nfs**	**nft**	**ni**
nsdiag	**nse**	**probe**	**pxp**
rlbdaemon	**sockregd**	**strlog**	**tcp**
timod	**tirdwr**	**udp**	

event_list takes the form:

event_spec [**,** *event_spec* **. . .**]

where *event_spec* takes one of the three forms:

[**!**] *integer* [**!**] *range* [**!**]*****

integer is an integer in hexadecimal (leading **0x**), octal (leading **0**), or decimal, which specifies a log event for the subsystem indicated.

range takes the form *integer* **-** *integer*, and indicates an inclusive set of events.

X25 Naming and Filtering

The X25 product provides capabilities to assign symbolic names to important numbers and to filter log events and trace messages. See *x25log*(1M) and *x25trace*(1M) for more information about X25 naming and filtering.

OTS Filtering

The OTS subsystem filter allows filtering of the message ID numbers that are typically found in the data portion of an OTS subsystem's log or trace record. The OTS subsystem filter is effective for any subsystem that is a member of the OTS subsystem group.

OTS trace filtering configuration commands have the following form in *config_file*:

OTS [*subsystem*] **msgid** [**!**] *message_ID* | *

Keywords and arguments are interpreted as follows:

OTS Identifies the filter as an OTS subsystem filter.

subsystem One of the following group of OTS subsystems:

OTS	**ACSE_PRES**	**NETWORK**
TRANSPORT	**SESSION**	

Note: The absence of *subsystem* implies that the filter applies to all OTS subsystems.

message_ID is the value of the message ID to filter. A message ID is used by OTS subsystems to identify similar types of information. It can be recognized as a 4 digit number contained in brackets (**[ ]**) at the beginning of an OTS subsystem's trace or log record. Initially all *message_ID*s are enabled for formatting. To format records with specific *message_ID*s, turn off all message IDs using the **!*** operator, then selectively enable the desired message IDs. Only one *message_ID* is allowed on each line. Multiple lines are "OR"ed together.

STREAMS Filtering

The STREAMS subsystem filter allows filtering on some fields of the messages logged by STREAMS modules and drivers. See *strlog*(7) for more information.

EXTERNAL INFLUENCES

International Code Set Support

Single- and multi-byte character code sets are supported in data. Single-byte character codesets are supported in filenames.

DEPENDENCIES

netfmt only recognizes subsystems and filters from products which have been installed and configured.

WARNINGS

The syntax that was used for the obsolete LAN trace and log options has been mixed with the syntax for the **netfmt** command such that any old options files can be used without any changes. The combination of syntax introduces some redundancy and possible confusion. The global filtering options have the string

formatter filter as the first two fields, while the LAN filtering options merely have the string **filter** as the first field. It is expected that the older LAN filtering options may change to become more congruent with the global filtering syntax in future releases.

The **nettl** and **netfmt** commands read the **/etc/nettlgen.conf** file each time they are executed. These commands will not operate if the file becomes corrupted (see *nettl*(1M) and *netfmt*(1M)).

DIAGNOSTICS

Messages describe illegal use of **netfmt** command and unexpected EOF encountered.

EXAMPLES

The first group of examples show how to use command line options.

1. Format the last 50 records in file **/var/adm/nettl.LOG00** (the default log file):

```
netfmt -t 50 -f /var/adm/nettl.LOG00
```

2. Use the follow option to send *all* log messages to the console (normally, only **DISASTER**-class log messages are sent to the console in console form):

```
netfmt -f  /var/adm/nettl.LOG00  -F  > /dev/console
```

3. Monitor all log messages in a **hpterm** window:

```
hpterm -e /usr/sbin/netfmt -F -f /var/adm/nettl.LOG00
```

4. Read file **/var/adm/trace.TRC1** for binary data and use **conf.file** as the filter configuration file:

```
netfmt -c conf.file -f /var/adm/trace.TRC1
```

The remaining examples show how to specify entries in the filter configuration file used with the **-c** option.

1. Tell **netfmt** to format only **INFORMATIVE**-class log messages coming from the **NS_LS_IP** subsystem between 10:31:53 and 10:41:00 on 23 November 1993.

```
formatter     filter    time_from     10:31:53   11/23/93
formatter     filter    time_through  10:41:00   11/23/93
formatter     filter    class         !*
formatter     filter    class         INFORMATIVE
formatter     filter    subsystem     !*
formatter     filter    subsystem     NS_LS_IP
```

2. Map hardware address to name(LAN):

```
name          node1         08-00-09-00-0e-ca
name          node3         02-60-8c-01-33-58
```

3. Format only packets from either of the above hardware addresses:

```
filter        source        08-00-09-00-0e-ca
filter        source        02-60-8c-01-33-58
```

4. Format all packets transmitted from the local node, **local**, to the remote node, **192.6.1.3**, which reference local TCP service ports **login** or **shell**, or remote UDP port **777**:

```
filter        ip_saddr      local
filter        ip_daddr      192.6.1.3
filter        tcp_sport     login
filter        tcp_sport     shell
filter        udp_dport     777
```

5. Format a TCP connection from local node **node2** to **192.6.1.3** which uses **node2** service port **ftp** and remote port **1198**.

```
filter        connection    node2:ftp    192.6.1.3:1198
```

6. Format all packets except those that use interface **lan0**:

```
filter        interface     ! lan0
```

7. Format all logged events for subsystem **ip**. No other events are formatted. (By default, all events are formatted):

```
filter          subsystem       ip    event       *
```

8. Format only event **5003** for subsystem **ip**. Format all events except **3000** for subsystem **tcp**. No other events are formatted.

```
filter          subsystem       ip     event  5003
filter          subsystem       tcp    event  *,!3000
```

9. Format only events **5003**, **5004**, **5005**, and **5006** for subsystem **ip**. Format all events except events **3000**, **3002**, and **3003** for subsystem **tcp**. No other events are formatted:

```
filter      subsystem       ip     event  5003-5006
filter      subsystem       tcp    event  *,!3000,!3002-3003
```

10. Format only those records containing message IDs **9973** and **9974** for subsystem **session** and those not containing message ID **9974** for subsystem **transport**. All records from other subsystems are formatted:

```
ots session    msgid          !*
ots session    msgid          9973
ots session    msgid          9974
ots transport  msgid          !9974
```

11. Combine LAN and general filtering options into one configuration file. Format 15 minutes of pduin and pduout data starting at 3:00 PM on 2 April 1990 for data from **lan0** interface.

```
formatter       filter        kind            0x30000000
formatter       filter        time_from       15:00:00 04/02/90
formatter       filter        time_through    15:15:00 04/02/90
filter          interface     !*
filter          interface     lan0
```

AUTHOR

netfmt was developed by HP.

FILES

/etc/nettlgen.conf	default subsystem configuration file
/var/adm/conslog.opts	default console logging options filter file
$HOME/.netfmtrc	default filter configuration file if the **-c config_file** option is not used on the command line.

SEE ALSO

nettl(1M), kl(1M), nettlconf(1M), nettlgen.conf(4), strlog(7).

NAME

nettl - control network tracing and logging

SYNOPSIS

/usr/sbin/nettl -start

/usr/sbin/nettl -stop

/usr/sbin/nettl -firmlog 0|1|2 -card *dev_name* ...

/usr/sbin/nettl -log *class* ... **-entity** *subsystem* ...

/usr/sbin/nettl -status [**log** |**trace** |**all**]

/usr/sbin/nettl -traceon *kind* ... **-entity** *subsystem* ... [**-card** *dev_name* ...]
[**-file** *tracename*] [**-m** *bytes*] [**-size** *portsize*] [**-tracemax** *maxsize*] [**-n** *num_files*]

/usr/sbin/nettl -traceoff -entity *subsystem* ...

DESCRIPTION

The **nettl** command is a tool used to capture network events or packets. Logging is a means of capturing network activities such as state changes, errors, and connection establishment. Tracing is used to capture or take a snapshot of inbound and outbound packets going through the network, as well as loopback or header information. A subsystem is a particular network module that can be acted upon, such as **ns_ls_driver**, or **X25L2**. **nettl** is used to control the network tracing and logging facility.

Except for the **-status** option, **nettl** can be used only by users who have an effective user ID of 0.

Options

nettl recognizes the following options, which can be used only in the combinations indicated in SYNOPSIS. Some option and argument keywords can be abbreviated as described below. All keywords are case-insensitive.

-start (Abbr.: **-st**)
Used alone without other options.

Initialize the tracing and logging facility, start up default logging, and optionally start up console logging. Logging is enabled for *all* subsystems as determined by the **/etc/nettlgen.conf** file. Log messages are sent to a log file whose name is determined by adding the suffix **.LOG00** to the log file name specified in the **/etc/nettlgen.conf** configuration file. Console logging is started if console logging has been configured in the **/etc/nettlgen.conf** file. See *nettlconf*(1M) and *nettlgen.conf*(4) for an explanation of the configuration file. If the log file (with suffix) already exists, it is opened in *append* mode; that is, new data is added to the file. The default name is **/var/adm/nettl** (thus logging starts to file **/var/adm/nettl.LOG00**). See "Data File Management" below for more information on how the log file is handled.

A **nettl -start** command is performed during system startup if the **NETTL** variable in the **/etc/rc.config.d/nettl** file has a value of **1**.

Note: It is strongly recommended that the tracing and logging facility be turned on before any networking is started and remain on as long as networking is being used. Otherwise, information about disasters will be lost. To minimize the impact on the system, all subsystems can be set with the **-log** option to capture only **disaster**-class log messages.

-stop (Abbr.: **-sp**)
Used alone without other options.

Terminate the trace/log facility. Once this command is issued, the trace/log facility is no longer able to accept the corresponding trace/log calls from the network subsystems.

Note: See note for the **-start** option.

-card *dev_name* ...
(Abbr.: **-c**)
This option is required by the X.25 subsystems; it is optional for other subsystems. Some subsystems do not support this option.

n

Limit the trace information gathered to only the data that comes from the specified network interface card. More than one *dev_name* can be specified at a time in order to trace multiple network interfaces.

dev_name specifies a device which corresponds to a network interface card that has been installed and configured. It can be either an integer representing the network interface, or the device file name of the network interface. Some subsystems do not support both types of *dev_name*. For example, the X25 subsystems require that *dev_name* be a device file name. The product documentation for the subsystems should explain if the **-card** option is applicable and how to choose an appropriate *dev_name*.

If *dev_name* is not an integer it is assumed to be a device file name. The path prefix **/dev/** will be attached in front of *dev_name* if it is not an absolute path name to form the device file name, **/dev/***dev_name*. *dev_name* must refer to a valid network device file.

-entity all
-entity *subsystem* ...
(Abbr.: **-e**)

Limit the action of **-log**, **-traceoff**, or **-traceon** to the specified protocol layers or software modules specified by *subsystem*.

The number and names of *subsystem*s on each system are dependent on the products that have been installed. Use the command **nettlconf -status** to obtain a full listing of supported subsystems and the products that own them.

Examples of OSI subsystems:

acse_pres	**ftam_init**	**mms**
asn1	**ftam_resp**	**network**
cm	**ftam_vfs**	**ots**
em	**ftp_ftam_gw**	**transport**
ftam_ftp_gw	**hps**	**ula_utils**

Examples of LAN subsystems:

ns_ls_driver	**ns_ls_loopback**	**ns_ls_ni**
ns_ls_icmp	**ns_ls_netisr**	**ns_ls_tcp**
ns_ls_igmp	**ns_ls_nfs**	**ns_ls_udp**
ns_ls_ip	**ns_ls_nft**	**ns_ls_x25**

Two X.25-specific subsystems are used for tracing only:

X25L2	**X25L3**

-file *tracename*
(Abbr.: **-f**)
Used with the first **-traceon** option only.

The first time the **-traceon** keyword is used, it initializes tracing, creating a file *tracename***.TRC0** which receives the binary tracing data. If a trace file of the name *tracename***.TRC0** already exists the binary trace data is appended to the end of the file.

To start a fresh trace file, first turn off tracing then turn it back on again using a different *tracename*. See "Data File Management" below for more information on file naming.

If **-file** is omitted, *binary* trace output goes to standard output. If standard output is a terminal device, an error message is issued and no tracing is generated.

-firmlog 0|1|2
(Abbr.: **-fm**)
Requires the **-card** option.
Series 800 and X.25 only.

Set the X.25/800 interface card logging mask to level 0, 1, or 2. The default level is 0. The X.25/800 interface logs a standard set of messages. A level of 1 specifies cautionary messages as well as the default messages. A level of 2 specifies information messages in addition to the cautionary and default messages. This option is recognized only by the **ns_ls_x25** subsystem.

-log *class* ... (Abbr.: **-l**)
Requires the **-entity** option.

Control the class of log messages that are enabled for the subsystems specified by the **-entity** option.

class specifies the logging class. Available classes are:

Full	Abbr.	Mask
informative	**i**	1
warning	**w**	2
error	**e**	4
disaster	**d**	8

informative	Describes routine operations and current system values.
warning	Indicates abnormal events possibly caused by subsystem problems.
error	Signals an event or condition which was *not* affecting the overall subsystem or network operation, but may have caused an application program to fail.
disaster	Signals an event or condition which *did* affect the overall subsystem or network operation, caused several programs to fail or the entire node to shut down.

Classes can be specified as keywords or as a single numeric mask depicting which classes to log. The mask is formed by adding the individual masks of the log classes. If you choose to indicate several classes at once, be sure to separate each log class with a space.

disaster logging is always on. The default logging classes for each subsystem is configured into the configuration file, **/etc/nettlgen.conf**. When the tracing/logging facility is started, the information in the configuration file is read and subsystems are enabled for logging with the specified classes. To change the log class, use the "**nettl -log** *class* **-entity** *subsystem*" command with a new log class value. If desired, the command can be run for different log classes and different entities.

-m *bytes* Specify the number of bytes (*bytes*) of each trace record to trace. This option allows the user to specify the number of bytes to be captured in the trace packet. The user may prefer not to capture an entire PDU trace, such as when the user is only interested in the header.

The maximum value for *bytes* is 2000. By default, the entire packet is traced. A value of 0 will also cause the entire packet to be traced. This option currently applies only to kernel subsystems.

-size *portsize*
(Abbr.: **-s**)
Used with first **-traceon** option only.

Set the size in kilobytes (KB) of the trace buffer used to hold trace messages until they are written to the file. The default size for this buffer is 68 KB. The possible range for *portsize* is 1 to 1024. Setting this value too low increases the possibility of dropped trace messages from fast subsystems.

-status log
-status trace
-status [**all**]
(Abbr.: **-ss**)
Used alone without other options.

Report the tracing and logging facility status. The facility must be operational, that is, **nettl -start** has been completed. The additional options define the type of trace or log information that is to be displayed. The default value is **all**.

log	Log status information
trace	Trace status information
all	Trace and log status information

-tracemax *maxsize*
(Abbr.: **-tm**)
Used with first **-traceon** option only.

Tracing uses a circular file method such that when one file fills up, another file is used. The number of trace files that can exist on a system at any given time can be specified using the **-n** option. See "Data File Management" below for more information on file behavior.

maxsize specifies the maximum size in kilobytes (KB) of any two trace files combined. Therefore, the maximum size of each trace file will be approximately half of *maxsize* kilobytes (KB). The default value for the combined file sizes is 1000 KB. The possible range for *maxsize* is 100 to 99999.

-n *num_files* Used with first **-traceon** option only.

Specifies the number of trace files that can exist on a system at any given time. However, nettl can reduce the number of trace files depending on the available disk space. If the option is not specified, the default value is two trace files.

-traceoff (Abbr.: **-tf**)
Requires the **-entity** option.

Disable tracing of *subsystem*s specified by the **-entity** option. If **all** is specified as an argument to the **-entity** option, all tracing is disabled. The trace file remains, and can be formatted by using the **netfmt** command to view the trace messages it contains (see *netfmt*(1M)).

-traceon all
-traceon *kind* ...
(Abbr.: **-tn**)
Requires the **-entity** option. The **-card** option is required for X.25 subsystems. Other options are not required.

Start tracing on the specified subsystems. The tracing and logging facility must have been initialized by **nettl -start** for this command to have any effect. The default trace file is standard output; it can be overridden by the **-file** option. If standard output is a terminal device, then an informative message is displayed and no trace data is produced.

When tracing is enabled, every operation through the subsystems is recorded if the *kind* mask is matched.

kind defines the trace masks used by the tracing facility before recording a message. If **-traceon all** is specified, all trace masks are enabled. *kind* can be entered as one or several of the following keywords or masks:

keyword	mask	keyword	mask
hdrin	**0x80000000**	**state**	**0x04000000**
hdrout	**0x40000000**	**error**	**0x02000000**
pduin	**0x20000000**	**logging**	**0x01000000**
pduout	**0x10000000**	**loopback**	**0x00800000**
proc	**0x08000000**		

hdrin	Inbound Protocol Header.
hdrout	Outbound Protocol Header.
pduin	Inbound Protocol Data Unit (including header and data).
pduout	Outbound Protocol Data Unit (including header and data).
proc	Procedure entry and exit.
state	Protocol or connection states.
error	Invalid events or condition.
logging	Special kind of trace that contains a log message.
loopback	Packets whose source and destination system is the same.

For multiple *kind*s, the masks can be specified separately or combined into a single number. For example, to enable both **pduin** and **pduout** (to trace all packets coming

n

into and out of the node) use either **pduin pduout** or **0x10000000 0x20000000** or the combination **0x30000000**.

Not all subsystems support all trace *kind*s. *No* error is returned if a given subsystem does not support a particular trace *kind*.

If a **-traceon** is issued on a subsystem that is already being traced, the tracing mask and optional values are changed to those specified by the new command, but the new **-file**, **-size**, and **-tracemax** options are ignored and a message is issued.

If **-entity all** is specified, all recognized subsystems are traced except X.25-specific subsystems. To turn on tracing for X.25, use the command

nettl -traceon *kind* **-e** *x.25_subsys* **-card** *dev_name*

where the value of *x.25_subsys* is **X25L2** or **X25L3**.

Data File Management

Data files created by the tracing and logging facility require special handling by the facility that the user must be aware of. When files are created, they have the suffix **.LOG00** or **.TRC00** appended to them, depending on whether they are log or trace files, respectively. This scheme is used to keep the files distinct for cases where the user specifies the same name in both places. Also, the files implement a type of circular buffer, with new data always going into the file appended with **.LOG00** or **.TRC00**. When a logname.LOG00 or tracename.TRC00 file is full, each log or trace is renamed to the next higher number in its sequence; that is, a file with sequence number **N** is renamed as a file with sequence number **N+1** and a new file named *logname***.LOG00** or *tracename***.TRC00** is created. The number of files that can exist simultaneously on the system can be specified by the **-n** option.

Note: The file name prefix (*logname* or *tracename*) specified by the user must not exceed eight characters so that the file name plus suffix does not exceed fourteen characters. Longer names are truncated. To see the actual name of the trace or log file, use the **nettl -status all** command.

Console Logging

Console logging is used to display significant log events on the system console. The values in the **/etc/nettlgen.conf** file determine if console logging is to be started and the entries in the **/var/adm/conslog.opts** file determine what log messages will be reported to the console. The **nettlconf** command can be used to configure and maintain the information in the **/etc/nettlgen.conf** file (see *nettlconf*(1M)). If changes are made to these files, **nettl** must be stopped and restarted for the new information to take effect.

All log messages written to the console as a result of this configuration information are in a special short form. If more information is desired on the console, the **netfmt** formatter can be used to direct output to the console device. This may be most useful in an X windows environment.

Console logging may be disabled if conservation of system resources is valued more than notification of log events.

n

EXTERNAL INFLUENCES

International Code Set Support

Single- and multibyte character code sets are supported in data; single-byte character code sets are supported in file names.

EXAMPLES

1. Initialize the tracing/logging facility:

   ```
   nettl -start
   ```

   (See note for the **-start** option.)

2. Display the status of the tracing/logging facility.

   ```
   nettl -status all
   ```

3. Change log class to **error** and **warning** for all the subsystems. **disaster** logging is always on for all subsystems.

   ```
   nettl -log e w -e all
   ```

4. Turn on inbound and outbound PDU tracing for the **transport** and **session** (OTS/9000) subsystems and send binary trace messages to file **/var/adm/trace.TRC0**.

```
nettl -traceon pduin pduout -entity transport session \
     -file /var/adm/trace
```

5. Turn on outbound PDU tracing for X.25 level two, and subsystem **ns_ls_ip**. Trace messages go to the trace file set up in the previous example. This example also uses the abbreviated options. Tracing for X.25 requires a **-card** option to indicate which X.25 card to trace.

```
nettl -tn pduout -e X25L2 ns_ls_ip -c x25_0
```

6. Determine status of tracing from the previous two examples.

```
nettl -status trace
```

The output should resemble the following:

```
Tracing Information:
Trace Filename:                  /var/adm/trace.TRC*
Trace file size(Kbytes): 1000
User's ID:               0       Buffer Size:            32768
Messages Dropped:        0       Messages Queued:        0

Subsystem Name:     Trace Mask:                     Card:
TRANSPORT           0x30000000
SESSION             0x30000000
NS_LS_IP            0x10000000
X25L2               0x10000000                      x25_0
```

7. Stop tracing for all subsystems.

```
nettl -traceoff -e all
```

8. Enable **pduin** and **pduout** tracing for **ns_ls_driver** (LAN driver) subsystem. Binary trace data goes to file **/var/adm/LAN.TRC0**.

   The **-file** option of this command is only valid the first time tracing is called. The trace file is not automatically reset with the **-file** option. To change the trace output file, stop tracing and start up again. This example assumes that the **-traceon** option is being used for the first time.

```
nettl -tn pduin pduout -e ns_ls_driver -file /var/adm/LAN
```

9. Terminate the tracing and logging facility.

```
nettl -stop
```

(See note for the **-start** option.)

WARNINGS

Although the **nettl** command allows the specification of all log classes and all trace kinds for all subsystems, many subsystems do not support all log classes and all trace kinds. No error or warning will be issued if a subsystem does not support a log class or trace kind. Refer to the product documentation of the subsystem for information on supported log classes and trace kinds.

Tracing to a file that resides on a NFS file system can impact system performance and result in loss of trace data. It is recommended that NFS file systems not be used to contain tracing output files.

Tracing to a file may not be able to keep up with a busy system, especially when extensive tracing information is being gathered. If some data loss is encountered, the trace buffer size can be increased. Be selective about the number of subsystems being traced, as well as the kinds of trace data being captured.

The **nettl** and **netfmt** commands read the **/etc/nettlgen.conf** file each time they are run (see *nettl*(1M) and *netfmt*(1M)). If the file becomes corrupted, these commands will no longer be operational.

FILES

/dev/netlog	Kernel log pseudo-device file.
/dev/nettrace	Kernel trace pseudo-device file.
/etc/nettlgen.conf	Tracing and logging subsystem configuration file.
/etc/rc.config.d/nettl	Contains variables which control the behavior of **nettl** during system startup.

n

/var/adm/conslog.opts	Default console logging options filter file as specified in **/etc/nettlgen.conf**.
/var/adm/nettl.LOG00	Default log file as specified in **/etc/nettlgen.conf**.

AUTHOR

nettl was developed by HP.

SEE ALSO

netfmt(1M), nettlconf(1M), nettlgen.conf(4).

NAME

nettladm - network tracing and logging administration manager

SYNOPSIS

/opt/nettladm/bin/nettladm [**-t**|**-l**] [**-c** *filter_file*]

DESCRIPTION

The **nettladm** command is a tool used to administer network tracing and logging. It provides an interactive user interface to the nettl, netfmt, and nettlconf commands. The interface runs in either text terminal mode or in a Motif graphical environment. To run **nettladm** using Motif windows set the **DISPLAY** environment variable to match the system name (e.g., **DISPLAY=***system***:0.0**) prior to using the command.

The **nettladm** command starts a menu-driven program that makes it easy to perform network tracing and logging tasks with only limited specialized knowledge of HP-UX. **nettladm** is a self-guided tool, and context-sensitive help is available at any point by pressing the f1 function key.

Options

nettladm recognizes the following options:

-l	Shortcut to enter the "Logging Subsystems" (logging) area. This is the default.
-t	Shortcut to enter the "Tracing Subsystems" (tracing) area.
-c *filter_file*	Use the contents of *filter_file* as the default set of subsystem formatting criteria when creating reports within the "Create Report" area. The defaults can be overridden through the interface screens. Global filters (those beginning with the word **FORMATTER**) and comments are ignored. See *netfmt*(1M) for the description and syntax of *filter_file*.

EXTERNAL INFLUENCES

International Code Set Support

Single- and multibyte character code sets are supported in data; single-byte character code sets are supported in file names.

WARNINGS

Changes to logging and tracing levels and states are not preserved across system reboots or stops and restarts from outside of the **nettladm** command. Permanent changes must be made to the **/etc/nettlgen.conf** file using the **nettlconf** command. Note that changes to console logging and all logging startup parameters are preserved.

Although the **nettladm** command allows the specification of all log classes and all trace kinds for all subsystems, many subsystems do not support all log classes and all trace kinds. No error or warning will be issued if a subsystem does not support a log class or trace kind. Refer to the product documentation of the subsystem for information on supported log classes and trace kinds.

The **nettladm** command reads the **/etc/nettlgen.conf** and **/var/adm/conslog.opts** files each time it is run (see *nettlgen.conf*(4)). If the files become corrupted, this command will no longer be operational.

DEPENDENCIES

nettladm runs in an X Windows environment as well as on the following kinds of terminals or terminal emulators:

- HP-compatible terminal with programmable function keys and on-screen display of function key labels.
- VT-100

FILES

/etc/nettlgen.conf	Tracing and logging subsystem configuration file.
/var/adm/conslog.opts	Default console logging options filter file as specified in **/etc/nettlgen.conf**.
/var/adm/nettl.LOG00	Default log file as specified in **/etc/nettlgen.conf**.
/var/adm/nettl.TRC0	Default trace file.

/opt/nettladm/lib/X11/app-defaults/Nettladm
X11 application defaults file.

AUTHOR

nettladm was developed by HP.

SEE ALSO

nettl(1M), netfmt(1M), nettlconf(1M), nettlgen.conf(4).

NAME

nettlconf - configure network tracing and logging command subsystem database

SYNOPSIS

/usr/sbin/nettlconf [**-KL**] **-status**

/usr/sbin/nettlconf -L [**-console** *conlog*] [**-portsize** *logportsize*] [**-space** *maxlogspace*] [**-filename** *logfilename*] [**-option** *logoptfile*]

/usr/sbin/nettlconf [**-KL**] [**-qmin** *minimumklqueuesize*] [**-qmax** *maximumklqueuesize*] [**-space** *maxlogspace*] [**-filename** *logfilename*] [**-write** *writelog*]

/usr/sbin/nettlconf [**-S**] **-id** *ssid* **-name** *ssname* [**-class** *logclass*] [**-kernel|-st[reams]**] **-lib** *sslib* **-msg** *ssmsgcat* [**-fmtfn** *fmtfunc*] [**-optfn** *optfunc*] **-group** *ssgrpname*

/usr/sbin/nettlconf -delete *ssid*

DESCRIPTION

nettlconf maintains the database file **/etc/nettlgen.conf** which contains information required by the **nettl**, **kl**, and **netfmt** commands (see *nettl*(1M), *kl*(1M), and *netfmt*(1M)). This database contains system logging information along with a description of each subsystem that uses either *NetTL* or *KL* facility to log messages.

nettlconf can be used to update the network or kernel logging parameters or to add, update and delete subsystem descriptions. If a subsystem already exists with the same *ssid*, the values given are substituted for those in the database; otherwise a new entry is created.

System administrators may use the **nettlconf** command to customize the network or kernel logging parameters stored in the database such as console logging behavior, the system log file name, the maximum system log file size, and the amount of memory required by *NetTL* and *KL* facilities.

nettlconf is also called during system startup to change the database to reflect the values of any relevant environment variables in the **/etc/rc.config.d/nettl** file.

Products use the **nettlconf** command during product installation to configure subsystems into the *NetTL* and *KL* facilities. The installation will execute the **nettlconf** command for each subsystem it installs in order to provide the information necessary for the subsystem to use the *NetTL* and *KL* facilities.

Only users with appropriate privileges can invoke **nettlconf** to modify the configuration file.

Options

The following options can be used to view the network or kernel logging parameters and all subsystem descriptions from the **nettlgen.conf** database.

-status (abbrev: **-s**) display the contents of the database relevant to the network logging facility only.

-KL -status display the contents of the database relevant to the kernel logging facility only.

The following options can be used to update configuration information about network logging.

-L This indicates that subsequent options apply to updating network logging information. Changes to logging information will not take effect until **nettl** has been stopped and restarted. This is a required field.

-console *conlog* (abbrev: **-c**) *conlog* is set to 1 if console logging is to be enabled when **nettl** is started, 0 if not. (Console logging is used to report interesting events on the system console.) This is an optional field.

NOTE: during system startup *conlog* will be changed to match the value of the *NETTL_CONSOLE* variable in the **/etc/rc.config.d/nettl** file.

-portsize *logportsize*
(abbrev: **-p**) *logportsize* determines the number of outstanding messages possible in the log queue. The value is in multiples of 1024 bytes. Valid range is 1 through 64. The default is 8. This is an optional field.

-space *maxlogspace*
(abbrev: **-s**) *maxlogspace* is the maximum logging file space to be allowed. This is

n

the combined size of the 2 ping-ponged log files. Specify the size in multiples of 1024 bytes. Valid range is 1 through 10240. Default is 1000. This is an optional field.

-filename *logfilename*
: (abbrev: **-f**) *logfilename* is the path and file name to be used as the system log file, without the ping-pong extension (.LOGx). The default system log file is **/var/adm/nettl**. This is an optional field.

-option *logoptfile*
: (abbrev: **-o**) *logoptfile* is the path and file name to be used as the console log options file. The information in this file will be used to select logged events that will be reported to the system console. The default console logging options file is **/var/adm/conslog.opts**. This is an optional field.

The following options can be used to update configuration information about kernel logging.

-KL
: This indicates that subsequent options apply to updating kernel logging information. Changes pertinent to the writing modules of the kernel loggin facility, such as kernel logging file name and maximum space for kernel log file (see below) will take effect whenever writing facility gets turned on. Changes to the kernel logging facility as a whole will not take effect until **kl** has been stopped and started.

-qmin *minimumklqueuesize*
: *minimumklqueuesize* determines the minimum number of outstanding messages possible in the log queue of *KL*. Valid range is 100 through 10000. The default is 1000. This is an optional field.

-qmax *maximumklqueuesize*
: *maximumklqueuesize* determines the maximum number of outstanding messages possible in the log queue of *KL*. Valid range is 100 through 10000. The default is 1000. This is an optional field.

-space *maxlogspace*
: (abbrev: **-s**) *maxlogspace* is the maximum logging file space to be allowed. This is the size of one ping-ponged log files. Valid range is 8192 (8K) through 1024M. Default if 1M. This is an optional field.

  Note: One can use suffixes **K** and **M** to specify whether size is meant to be in Kilo or Mega bytes.

-filename *logfilename*
: (abbrev: **-f**) *logfilename* is the path and file name to be used as the kernel log file, without the ping-pong extension (**.KLOGx**). The default system kernel log file is **/var/adm/kl**. This is an optional field.

-write *writelog*
: (abbrev: **-w**) *writelog* is set to 1 if writing kernel log to disk is to be enabled when **kl** is started, 0 if not. Default is 0. This is an optional field.

The following options are used to add or update a subsystem description to the database.

-S
: Indicates that subsequent options apply to adding or updating a subsystem entry. This is an optional field.

-id *ssid*
: (abbrev: **-i**) *ssid* (subsystem ID number) is used as the key field in the **nettlgen.conf** database. It uniquely identifies a subsystem to the *NetTL* and *KL* facilities. This is a required field.

  Note: Subsystems are mutually exclusively supported by *NetTL* and *KL* facilities. This means that a given subsystem logs its messages either through *NetTL* or *KL*, but not both. The following rule applies: if the subsystem ID number is within 0 through 511 (including end numbers) range, then *NetTL* facility takes care of those messages; if the subsystem ID number is within 512 through 1023 (including end numbers) range then *KL* facility takes care of those messages. Subsystem ID numbers greater than 1023 are not allowed.

-name *ssname*
: (abbrev: **-n**) *ssname* is the subsystem-name mnemonic. This string is used to identify the subsystem on the **nettl** and **kl** command lines and also in the subsystem header displayed by the formatter (see *nettl*(1M), *kl*(1M) and *netfmt*(1M)). This is a required field.

-class *logclass* (abbrev: **-c**) *logclass* is the default log class mask assigned to the subsystem at start-up of *NetTL* or *KL* facility. This is an optional field.

There is an important difference between the interpretation of the logclass by *NetTL* and *KL* facilities.

Interpretation By NetTL

For multiple classes, the masks must be combined into a single decimal number. For example, to initially log **DISASTER** and **ERROR** events use **12** as the *logclass*. Default is an empty field in **nettlgen.conf**. **nettl** substitutes **12** (disaster and error) for an empty class field.

Interpretation By KL

For *KL*, the following rule applies for the messages to log: if level x is specified then all messages whose severity is greater or equal to the severity of class x will be logged. For example, if the logclass is 2, all warning, error and disaster messages will be logged by *KL*. Default is an empty field in **nettlgen.conf**. **kl** substitutes 8 (disaster) for an empty class field.

Class	Abbreviation
informative	1
warning	2
error	4
disaster	8

-kernel (abbrev: **-k**) flags the given subsystem as a kernel subsystem. **nettl** uses this information to control certain tracing and logging properties of the subsystem. If a subsystem is serviced by *NetTL* facility, then it is defaulted to non-kernel unless this option is specified, whereas any subsystem serviced by *KL* facility is defaulted as a kernel unless otherwise specified. This is an optional field.

-streams (abbrev: **-st**) flags the given subsystem as a streams based kernel subsystem. **nettl** uses this information to control certain tracing and logging properties of the subsystem. A subsystem is defaulted to non-kernel unless this option is used. This is an optional field.

n

-lib *sslib* (abbrev: **-l**) *sslib* is the name of the shared library where the subsystem formatter resides. This should be an absolute path name unless the library resides in **/usr/lib**. Multiple subsystems can reference the same library. This is a required field.

-msg *ssmsgcat* (abbrev: **-m**) *ssmsgcat* is the name of the subsystem formatter message catalog. If the pathname and **.cat** filename extension are excluded, */usr/lib/nls/%L/%N.cat* is used to locate *ssmsgcat*. Otherwise, *ssmsgcat* must be formatted similarly to the **NLSPATH** environment variable (see *environ*(5)). Multiple subsystems can refer to the same message catalog. This is a required field.

-fmtfn *fmtfunc* (abbrev: **-f**) *fmtfunc* specifies the function to call when formatting data from the given subsystem. Multiple subsystems can reference the same formatting function. Default is to form the function name from the subsystem ID as follows:

subsys_*N***_format**

where *N* is the subsystem *ID* number. If a null function is needed for this subsystem, specify

-f NULL

This is an optional field.

-optfn *optfunc* (abbrev: **-o**) *optfunc* specifies the function used to process options in the **netfmt** filter configuration file (see *netfmt*(1M)). Multiple subsystems can reference the same options processing function. The default is an empty field in **nettlgen.conf**. **netfmt** assumes a *NULL* function for an empty *optfunc* field. This is an optional field.

-group *ssgrpname*
(abbrev: **-g**) *ssgrpname* is a group name associated with the subsystem. It is

typically the product name of the subsystem. Several subsystems can be grouped together so that a common banner is printed in the formatted header. This is a required field.

The following option is used to remove a subsystem description from the database.

-delete *ssid* (abbrev: **-d**) Deletes all information associated with the *ssid* (subsystem ID) from the database.

WARNINGS

The **nettlconf** utility is intended primarily for use by HP subsystems to configure themselves into the *NetTL* and *KL* facilites at installation time. System administrators may wish to use this command to alter the default logging class each subsystem starts up with, but no other information about the subsystem should be changed.

The **nettl**, **kl**, and **netfmt** commands read the **/etc/nettlgen.conf** file each time they are executed. If the file becomes corrupted these commands cannot function.

Some changes to the **/etc/nettlgen.conf** file do not take effect until **nettl**, **kl**, and **netfmt** are stopped and restarted.

AUTHOR

nettlconf was developed by HP.

FILES

/etc/nettlgen.conf	subsystem configuration file maintained by **nettlconf**
/etc/rc.config.d/nettl	configuration file controlling **nettl** during system startup

SEE ALSO

netfmt(1M), nettl(1M), kl(1M), nettlgen.conf(4), environ(5).

NAME

newaliases - rebuilds the database for the mail aliases file

SYNOPSIS

newaliases [**-on**]

DESCRIPTION

newaliases rebuilds the random access database for the mail aliases file **/etc/mail/aliases**. It must be run each time this file is changed in order for the change to take effect.

newaliases is identical to **sendmail -bi**.

Options

-on Validate addresses. When **sendmail** rebuilds the alias database files, it will check the legality of all addresses to the right of the colons. Each address is validated. If the validation fails, the address is skipped and a warning message is displayed.

RETURN VALUE

The **newaliases** utility exits 0 on success, and >0 if an error occurs.

AUTHOR

newaliases was developed by the University of California, Berkeley, and originally appeared in 4.0BSD. The manual page originally came from **sendmail 8.7**.

FILES

/etc/mail/aliases The mail aliases file.

/etc/mail/aliases.db
Database of alias names.

SEE ALSO

aliases(5), sendmail(1M).

NAME

newarray - configure a disk array

SYNOPSIS

newarray [**-N***Config_Name* | -r*RAID_Level*] [**Options**] *device_file*

DESCRIPTION

newarray, a front-end program for the utility **cfl** (see *cfl*(1M)), facilitates the configuration of Hewlett-Packard SCSI disk arrays. It is the recommended utility for all array configuration. Array configuration maps a set of one or more physical disk mechanisms in an array to a set of one or more logical disks, addressable by HP-UX. Logical disks are addressed through device files. Each logical disk in an array (also known as a LUN, for Logical UNit), has its own device file. A logical disk can consist of a single physical disk, a portion of a single physical disk, multiple physical disks, or portions of multiple physical disks. For additional information about possible array configurations, see the array configuration table contained in the file **/etc/hpC2400/arraytab**, and *arraytab*(4).

Supported configurations for the array device are pre-defined in the array configuration table, located in file **/etc/hpC2400/arraytab**.

newarray can configure a complete set of logical partitions for an array in one operation. Due to the inter-dependency of logical partitions, this is the recommended method for configuration. A single logical partition can be added to an array configuration using an entry from the array configuration table by using the **-L** option.

device_file is a character device file that specifies the I/O address, and driver to use when configuring the disk array. The way that this file is used by **newarray** is system dependent. See dependencies below. Logical partitions in an array are independently addressable by using the appropriate device file to address the logical unit assigned to a partition.

Prior to configuring the array (except with the **-L** option), all currently configured logical partitions are removed from the configuration.

To simplify array configuration **newarray** obtains much of the necessary information directly from the array device, and its attached disk mechanisms. The array model number, and the number of available physical disks available, is determined by querying the device. This information is used to locate the appropriate configuration entry in the array configuration table. Optional parameters can be used to override the default, and inquiry values.

The preferred configuration method is to use the **-N** option to specify a configuration by name. The name determines which configuration **newarray** uses from the array configuration table. Configuration parameters are obtained from the named configuration entry. Parameters of the chosen configuration can be overridden using options to **newarray**, or by creating and using a custom configuration entry in the array configuration table. See the WARNINGS section of this manpage.

Because the array controller type, and disk mechanism types are used in addition to the configuration name to select an entry from the array configuration table the configuration name does not have to be unique within the array configuration table. However, the combination of configuration name, array controller type, and disk mechanism types must be unique within the array configuration table. During configuration, the array controller type, and disk mechanism types are obtained by querying the devices.

The **-r** option specifies an operating mode, rather than specifying a configuration by name. The **-d** option, which specifies the size of a disk group, is often used with the **-r** option. If **-d** is not used, **newarray** selects the configuration in the array configuration table that most closely matches the disks in the array.

When the configuration parameters have been determined, **newarray** calls **cfl.**

If the **-V** option is used, **newarray** prints its actions, and the parameters it passes to **cfl** to configure the array (see *cfl*(1M)).

Array Configuration

newarray obtains its configuration values from the array configuration table. If not specified there, default values are provided by **cfl** (see *cfl*(1M)). Configuration values can be overridden by **newarray** options.

Options

-L *unit addr* Configures a single LUN from the specified configuration. The **-L** option is useful for adding disks to an array without changing the existing configuration. Because the order in

n

which LUN's are configured determines the physical mapping on the disks within the array, be very careful when using the **-L** option.

-N *config_name*
The name of the configuration to be used, as specified in the configuration file **/etc/hpC2400/arraytab**. See *arraytab(4)*.

-V Display the parameters of array configuration, and the utility commands issued as part of the configuration process.

-b *block_size* The size in bytes of the LUN block. Must be an integral number of the physical disk mechanism sector size. Currently supported values are 512, 1024, 2048, and 4096.

-c *capacity* The size in blocks of the LUN. A value of 0 defaults to the largest capacity available. If the LUN type is set to sub-LUN, the capacity is the available capacity of the composite drive group or 2 GByte if the 2 GByte flag is set, which ever is smaller. See **-f** option.

-d *group_size* Physical drive group will contain this number of disks in the logical partition configuration.

-f *flags* Configuration flags. There are 16 flags, represented by a 16 bit hexadecimal number. Currently only four of the flags are defined. The flag definitions and their default value are:

Bit 0	*off*	Not used.
Bit 1	*on*	Disable auto reconstruction. When set (on), disables the automatic detection, and initiation of failed disk data reconstruction.
Bit 2	*off*	Not used.
Bit 3	*off*	Not used.
Bit 4	*on*	When set (on), enables AEN (automatic event notification) polling.
Bit 5	*on*	When set (on), enables read parity verification.
Bit 6	*on*	When set (on), enables write with parity verification.
Bit 7	*off*	Not used.
Bit 8	*off*	Mode Sense default pages. Bit 8 and Bit 9 concurrently set is reserved.
Bit 9	*off*	Mode Sense current pages. Bit 8 and Bit 9 concurrently set is reserved.
Bit 10-15	*off*	Not used.

-g *group_name*
Use physical drive group configuration with label GroupName (in array configuration table) for this LUN configuration.

-i *seg0_size* The size in bytes of the first segment LUN. This allows this area to be set to a size different than the remainder of the disk, an area typically used as the boot block for some systems. This must be a integral number of the block-size. If there are no special requirements, this parameter should be set to 0.

-k *recon_size* Reconstruction size. The **-k** option specifies (in LUN blocks) the amount of data to be reconstructed in a single operation during reconstruction of a redundant drive configuration. Larger values provide more efficient (faster) reconstruction, but hold off the servicing of I/O requests. Smaller values allow quicker servicing of I/O requests, but with less efficient (slower) reconstruction.

-l *recon_freq* Reconstruction frequency. The **-l** option specifies (in tenths of a second) the time period between reconstruction of disk segments in a redundant drive configuration. Small time periods cause the array to consume most of its time reconstructing data, but allow the reconstruction to complete more quickly. Large time periods allocate more time to I/O processing, but require longer reconstruction times.

-r *raid_level* The RAID (redundancy level) to apply to the disks in the array. Valid entries for *raid_level* are RAID_0, RAID_1, RAID_3, and RAID_5. Some RAID levels require specific physical drive configurations. See also the **-g** option.

-s *seg_size* The number of bytes of a contiguous segment of the logical address space residing on a single physical disk. This affects how many physical disks are involved in a single I/O request. If I/O requests are mostly random, single-block requests, set this value to the integral number of the LUN block size that minimizes the number of disks necessary to service most

I/O requests. A larger size will allocate more time to I/O processing.

-t *LUN_type* LUNs can be configured as regular LUNs (**reg**), or sub-LUNs (**sub**). A regular LUN utilizes all the available capacity of a disk group, or limits the LUN configuration to 2 GBytes if the 2 GByte limiter is set. If a regular LUN configuration is used, the **-c** option is ignored. A sub-LUN allows logical partitioning of the disk group capacity into a maximum of eight LUNs. Valid values for *LUN_type* are "**reg**" and "**sub**".

Custom Configurations

You can create array configurations that might be better suited to a particular application by using **newarray**'s command line parameters to override default values, or by creating special entries in the array configuration table in the file **/etc/hpC2400/arraytab**. Before you do, see cautionary notes in the WARNINGS section of this manpage.

RETURN VALUES

newarray will return the following values:

0 Successful completion.
-1 Command failed (an error occurred).

ERRORS

```
newarray: device busy
```

To ensure that **newarray** does not modify a disk array that is being used by another process, **newarray** attempts to obtain exclusive access to the disk array. If the disk array is already opened by another process (for example, LVM — the Logical Volume Manager), a “**device busy**” error message is returned by the driver. To eliminate the “**device busy**” condition, determine what process has the device open. In the case of LVM, it is necessary to deactivate the volume group containing the array before configuring the array (see *vgchange*(1M)).

EXAMPLES:

The following examples use configurations contained in **/etc/hpC2400/arraytab**.

Raid Level Specification

To configure an HP C2425D with 5 internal disks to a five drive RAID level 0 configuration (on Series 700 computer):

```
newarray -rRAID_0 /dev/rdsk/c2t3d0
```

To configure an HP C2425D with 5 internal disks to a one drive RAID level 0 configuration (on Series 700):

```
newarray -rRAID_0 -d1 /dev/rdsk/c2t3d0
```

Name Specification

To configure an HP C2430D with five disks connected on SCSI channel 3 (on a Series 800) using the configuration "Raid_3_5d" in **/etc/hpC2400/arraytab**:

```
newarray -NRaid_3_5d /dev/rdsk/c2t3d0
```

WARNINGS

We strongly recommend that you use the array configurations that are specified, and delivered by Hewlett-Packard, in the file **/etc/hpC2400/arraytab.** These configurations have been tested and certified for proper use on Hewlett-Packard computer systems. Custom configurations cannot be warranted for proper operation.

Configuring a disk array causes the loss of user data on the array.

When using the **-L** option, physical media is assigned to the logical unit in the order in which the logical units are configured. Existing logical unit configurations are NOT removed prior to configuration with this option. The use of this option is not recommended at this time.

DEPENDENCIES

File System Considerations

The disk array maps the address space of one or more physical disk mechanisms onto logical "disk" partitions. The parameters defined in the configuration, together with the data access patterns of the user's application, determine the operating characteristics of the logical disk. Some configurations create multiple logical partitions, that share a set of physical disks. I/O traffic to each of the logical partitions affects performance, due to the common physical disk resources. The file system or application using the "logical" disk may require or assume certain characteristics. For optimal system performance it is necessary that the file system configuration and application be compatible with the array configuration.

Your choice of segment size directly affects the performance of the disk array. Choose this parameter in concert with the choice of the parameters used when building the file system on the device. In general, the segment size determines how much data from a single I/O will be stored on a single disk within the array. A smaller value will involve more of the disks with the I/O, whereas a larger value will involve fewer disks. If input/output operations tend to be very long, the involvement of multiple disks may hasten the completion of each I/O. In this case the access time is the same as a single disk, but the disk data transfer time is shared across the set of disks. If input/output operations are short, the access time will dominate relative to the disk data transfer time, and more input/output operations may be processed in parallel by involving fewer disks in each I/O. In all cases the relative locality of data and the access pattern will affect the performance. For highly sequential data, it may be advantageous to locate the data for a single I/O on a single disk, to take advantage of read-ahead caching within each disk.

Configurations for the HP C2430 disk array should enable the automatic data reconstruction LUN flag as part of the configuration specification.

Supported Array Products:

The HP C2425 and HP C2427 disk arrays are only supported on Series 700 systems running HP-UX version 9.0X.

The HP C2430 disk array is supported on Series 700 and 800 systems running HP-UX versions 9.0X and 10.0X.

AUTHOR

newarray was developed by HP.

SEE ALSO

arraytab(4), cfl(1M), buildfs(1M), fs(4), mkfs(1M), sss(1M), dcc(1M).

NAME

newfs - construct a new file system

SYNOPSIS

/usr/sbin/newfs [**-F** *FStype*] [**-o** *specific_options*] [**-V**] *special*

DESCRIPTION

The **newfs** command is a "friendly" front-end to the **mkfs** command (see *mkfs*(1M)). The **newfs** command calculates the appropriate parameters and then builds the file system by invoking the **mkfs** command.

special represents a character (raw) special device.

Options

newfs recognizes the following options:

-F *FStype* Specify the file system type on which to operate (see *fstyp*(1M) and *fs_wrapper*(5)). If this option is not included on the command line, then the file system type is determined from the file **/etc/fstab** by matching *special* with an entry in that file. If there is no entry in **/etc/fstab**, then the file system type is determined from the file **/etc/default/fs**.

-o *specific_options*
Specify options specific to the file system type. *specific_options* is a list of suboptions and/or keyword/attribute pairs intended for an *FStype*-specific module of the command. See the file system specific manual entries for a description of the *specific_options* that are supported, if any.

-V Echo the completed command line, but perform no other actions. The command line is generated by incorporating the specified options and arguments and other information derived from **/etc/fstab**. This option allows the user to verify the command line.

EXAMPLES

Execute the **newfs** command to create an HFS file system on **/dev/rdsk/c1t0d2**

```
newfs -F hfs /dev/rdsk/c1t0d2
```

AUTHOR

newfs was developed by HP and the University of California, Berkeley.

FILES

/etc/default/fs	File that specifies the default file system type.
/etc/fstab	Static information about the file systems.

SEE ALSO

fsck(1M), fstyp(1M), mkfs(1M), newfs_hfs(1M), newfs_vxfs(1M), fstab(4), fs_wrapper(5).

NAME

newfs - construct a new HFS file system

SYNOPSIS

/usr/sbin/newfs [**-F hfs**] [**-B**] [**-d**] [**-L**|**-S**] [**-O** *disk_type*] [**-R** *swap*] [**-v**] [**-V**]
[*mkfs-options*] *special*

DESCRIPTION

The **newfs** command builds a file system by invoking the **mkfs** command.

The **newfs** command creates the file system with a rotational delay value of zero (see *tunefs*(1M)).

special represents a character (raw) special device.

Options

newfs recognizes the following options:

-F hfs Specify the HFS file system type.

-B Reserve space for boot programs past the end of the file system. If file **/usr/lib/uxbootlf** is present on the system then sufficient space to accommodate that file is reserved, otherwise 691 KB sectors are reserved. This option decreases the size of the file system to be created. This option cannot be used if the **-s** option is given; see "mkfs Options" below.

-d This option allows the **newfs** command to make the new file system in an ordinary file. In this case, *special* is the name of an existing file in which to create the file system. The **-s** option (see "mkfs Options") must be provided with this option.

-L|**-S** There are two types of HFS file systems, distinguished mainly by directory formats that place different limits on the length of file names.

If **-L** is specified, build a long-file-name file system that allows directory entries (file names) to be up to **MAXNAMLEN** (255) bytes long.

If **-S** is specified, build a short-file-name file system that allows directory entries (file names) to be up to **DIRSIZ** (14) bytes long.

If neither **-L** nor **-S** is specified, build a file system of the same type as the root file system.

-O *disk_type* Use disk parameters from the entry for the named disk type in **/etc/disktab**. This option is provided for backward compatibility with previous HP-UX releases. Any parameters specified in the command line will override the corresponding values in **/etc/disktab**. Any values not given in the command line or in **/etc/disktab** will be defaulted.

-R *swap* Reserve *swap* megabytes (MB) of swap space past the end of the file system. This option decreases the size of the file system to be created by the given amount. This option cannot be used if the **-s** option is given; see "mkfs Options" below.

-v Verbose; the **newfs** command prints out its actions, including the parameters passed to the **mkfs** command.

-V Echo the completed command line, but perform no other actions. The command line is generated by incorporating the user-specified options and other information derived from **/etc/fstab**. This option allows the user to verify the command line.

Both the **-R** and **-B** options can be given in the same command line. In this case, both the requested swap space and the space needed for boot programs are reserved. These options are for use when the file system size defaults to the size of the entire disk.

mkfs Options

The *mkfs-options* argument can be zero or more of the following options that can be used to override default values passed to the **mkfs** command:

-b *blksize* The primary block size for files on the file system. Valid values are: 4096, 8192, 16384, 32768, and 65536. The default value is 8192 bytes.

-c *cylinders_per_group*
The number of disk cylinders per cylinder group. This number must be in the range 1

n

to 32. The default value is 16 cylinders per group.

-f *fragsize* The fragment size for files on the file system. *fragsize* represents the smallest amount of disk space to be allocated to a file. It must be a power of two no smaller than **DEV_BSIZE** and no smaller than one-eighth of the file system block size. The default value is 1024 bytes.

-i *number_of_bytes_per_inode*
The density of inodes in the file system specified as the number of bytes per inode. The default is 6144 bytes per inode.

This number should reflect the expected average size of files in the file system. If fewer inodes are desired, a larger number should be used; if more inodes are desired, a smaller number should be used.

Note: The number of inodes that will be created in each cylinder group of a file system is approximately the size of the cylinder group divided by the number of bytes per inode, up to a limit of 2048 inodes per cylinder group. If the size of the cylinder group is large enough to reach this limit, the default number of bytes per inode will be increased.

-m *free_space_percent*
The minimum percentage of free disk space allowed. The default value is 10 percent.

Once the file system capacity reaches this threshold, only users with appropriate privileges can allocate disk blocks.

-r *revolutions_per_minute*
The disk speed in revolutions per minute (rpm). The default value is 3600 revolutions per minute.

-s *size* The number of **DEV_BSIZE** blocks in the file system. **DEV_BSIZE** is defined in **<sys/param.h>**. The default value is the size of the entire disk or disk section minus any swap or boot space requested. See *mkfs_hfs*(1M) for limits on the size of HFS file systems.

n

-t *tracks_per_cylinder*
The number of tracks per cylinder. The default value depends on the size of the file system. For file systems of less than 500 MB, the default is 7; for file systems between 500 MB and 1 GB, the default is 12; for file systems larger than 1 GB the default is 16.

-o *specific_options*
Specify a list of comma separated suboptions and/or keyword/attribute pairs from the list below.

largefiles |**nolargefiles**
Controls the **largefile featurebit** for the file system. The default is **nolargefiles**. This means the bit is not set and files created on the file system will be limited to less than 2 gigabytes in size. If **largefiles** is specified, the bit is set and the maximum size for files created on the file system is not limited to 2 gigabytes (see *mount_hfs*(1M) and *fsadm_hfs*(1M)).

Access Control Lists

Every file with one or more optional ACL entries consumes an extra (continuation) inode. If you anticipate significant use of ACLs on a new file system, you can allocate more inodes by reducing the value of the argument to the **-i** option appropriately. The small default value typically causes allocation of many more inodes than are actually necessary, even with ACLs. To evaluate the need for extra inodes, run the **bdf -i** command on existing file systems. For more information on access control lists, see *acl*(5).

EXAMPLES

Execute the **newfs** command to create an HFS file system on a non-LVM disk **/dev/rdsk/c1t0d2** and reserve 40 megabytes of swap space.

```
newfs -F hfs -R 40 /dev/rdsk/c1t0d2
```

Create an HFS file system within a logical volume, **my_lvol**, whose size is identical to that of the logical volume. (Note the use of the character (raw) special device.)

```
newfs -F hfs /dev/vg01/rmy_lvol
```

WARNINGS

The old **-F** option, from prior releases of *newfs*(1M), is no longer supported.

newfs(1M) cannot be executed specifying creation of a file system on a whole disk if that disk was previously used as an LVM disk. If you wish to do this, use *mediainit*(1) to reinitialize the disk first.

AUTHOR

newfs was developed by HP and the University of California, Berkeley.

FILES

/etc/disktab	
/etc/fstab	Static information about the file systems.

SEE ALSO

bdf(1M), fsadm_hfs(1M), mkboot(1M), mkfs(1M), mkfs_hfs(1M), mount_hfs(1M), newfs(1M), tunefs(1M), disktab(4), fs(4), acl(5).

NAME

newfs - create a new VxFS file system

SYNOPSIS

/usr/sbin/newfs [**-F vxfs**] [**-B**] [**-O** *disk_type*] [**-R** *swap*] [**-V**] [**-v**]
[*mkfs_vxfs_options*] *special*

DESCRIPTION

newfs -F vxfs builds a VxFS file system by invoking **mkfs**. *special* specifies a character (or raw) file (for example, **/dev/rdsk/c0t6d0**).

Options

newfs recognizes the following options:

-F vxfs	Specify the file-system type **vxfs**.
-B	Reserve space for boot programs past the end of the file system. If the file **/usr/lib/uxbootlf** is present on the system, sufficient space to accommodate that file is reserved; otherwise 691 kilobytes are reserved. This option decreases the size of the file system being created. This option cannot be used if the file-system size is also specified using **-s** (see **mkfs_vxfs** options below).
-O *disk_type*	Use disk parameters from the *disk_type* named in **/etc/disktab**. This option is provided for backward compatibility with previous HP-UX releases. Any parameters specified on the command line will the corresponding values in **/etc/disktab**. Any values not specified in the command line and not shown in **/etc/disktab** are defaulted.
-R *swap*	Reserve *swap* megabytes of swap space past the end of the file system. This option decreases the size of the file system to be created by the specified number of megabytes. **-R** cannot be used if the file-system size is also specified using **-s** (see **mkfs_vxfs** options below).
-V	Echo the completed command line, but do not execute the command. The command line is generated by incorporating the user-specified options and other information derived from **/etc/fstab.** This option allows the user to verify the command line.
-v	Specify verbose mode. **newfs** prints out its actions, including the parameters passed to **mkfs**.

NOTE: You can specify the **-R** and **-B** options in the same command line. In that case, both the requested swap space and the space needed for boot programs are reserved. These options are used when the file system size is defaulted to the size of the entire disk.

mkfs_vxfs Options

The following additional command-line options can be used to override default parameters passed to **mkfs_vxfs:**

-b *block_size* File system block size in bytes. The default value is 1024 bytes.

-o largefiles | **nolargefiles**
Valid only for the Version 3 and later disk layouts. This option controls the largefiles flag for the file system. If **largefiles** is specified, the bit is set and files two gigabytes or larger can be created. If **nolargefiles** is specified, the bit is cleared and files created on the files system are limited to less than two gigabytes. The default is **nolargefiles**. See *mkfs_vxfs*(1M), *mount_vxfs*(1M) and *fsadm_vxfs*(1M).

-s *size* File system size in **DEV_BSIZE** blocks (defined in **<sys/param.h>**). The default value is the size of the entire disk or disk section, minus any swap or boot space requested. The *size* specifies the number of sectors in the file system. By default, size is specified in units of **DEV_BSIZE** sectors. However, you can append a suffix to *size* to indicate another unit of measure. Append **k** or **K** to indicate that the value is in kilobytes, **m** or **M** to indicate megabytes, or **g** or **G** to indicate gigabytes.

EXAMPLES

To create a VxFS file system on **/dev/rdsk/c1t5d0** and reserve 40 megabytes of swap space.

```
newfs -F vxfs -R40 /dev/rdsk/c1t5d0
```

FILES

`/etc/disktab`	Disk description file.
`/etc/fstab`	Static information about the file systems.

SEE ALSO

fsadm_vxfs(1M), mkfs(1M), mkfs_vxfs(1M), mount_vxfs(1M), newfs(1M), disktab(4).

NAME

newkey - create a new Diffie-Hellman key pair in the publickey database

SYNOPSIS

newkey -h *hostname* [**-s nisplus** | **nis** | **files**]

newkey -u *username* [**-s nisplus** | **nis** | **files**]

DESCRIPTION

newkey establishes new public keys for users and machines on the network. These keys are needed when using secure RPC or secure NFS service.

newkey prompts for a password for the given *username* or *hostname* and then creates a new public/secret Diffie-Hellman 192 bit key pair for the user or host. The secret key is encrypted with the given password. The key pair can be stored in the **/etc/publickey** file, the NIS **publickey** map, or the NIS+ **cred.org_dir** table.

newkey consults the **publickey** entry in the name service switch configuration file (see *nsswitch.conf*(4)) to determine which naming service is used to store the secure RPC keys. If the **publickey** entry specifies a unique name service, **newkey** will add the key in the specified name service. However, if there are multiple name services listed, **newkey** cannot decide which source to update and will display an error message. The user is required to specify the source explicitly with the **-s** option.

In the case of NIS, **newkey** should be run by the superuser on the master NIS server for that domain. In the case of NIS+, **newkey** should be run by the superuser on a machine which has permission to update the **cred.org_dir** table of the new user/host domain.

In the case of NIS+, *nisaddcred*(1M) should be used to add new keys.

Options

-h *hostname* Create a new public/secret key pair for the privileged user at the given *hostname*. Prompts for a password for the given *hostname*.

-u *username* Create a new public/secret key pair for the given *username*. Prompts for a password for the given *username*.

-s nisplus
-s nis
-s files Update the database in the specified source: **nisplus** (for NIS+), **nis** (for NIS), or **files**. Other sources may be available in the future.

AUTHOR

newkey was developed by Sun Microsystems, Inc.

SEE ALSO

chkey(1), keylogin(1), nisaddcred(1M), nisclient(1M), nsswitch.conf(4), publickey(4).

NAME

nfsd, biod - NFS daemons

SYNOPSIS

/usr/sbin/nfsd [**-a**] [**-p** *protocol*] [**-t** *device*] [*nservers*]

/usr/sbin/biod [*nservers*]

DESCRIPTION

nfsd starts the NFS server daemons that handle client file system requests (see *nfs*(7)). *nservers* is the suggested number of file system request daemons that will start. The minimum number of daemons will be equal to the number of active processors plus one, or to a multiple of the number of active processors greater than or equal to *nservers* plus one. To obtain the best performance in most cases, set *nservers* to at least sixteen.

biod starts *nservers* asynchronous block I/O daemons. This command is used on an NFS client to buffer cache handle read-ahead and write-behind. *nservers* is a number greater than zero. For best performance, set *nservers* to at least sixteen.

Options

nfsd recognizes the following options:

-a Start a NFS daemon over all supported connectionless and connection-oriented transports, including udp and tcp. The NFS_TCP environment variable in **/etc/rc.config.d/nfsconf** configuration file, must have been set to 1 (one) in order to support connect-oriented (tcp) transport.

-p *protocol*
Start a NFS daemon over the specified protocol.

-t *device* Start a NFS daemon for the transport specified by the given device.

nservers *nservers* is the suggested number of file system request daemons that will start. The actual number of daemons started will be one tcp daemon (to support kernel tcp threads) plus the number of udp daemons. The minimum number of udp daemons will be equal to a multiple of the active processors, or to *nservers*, whichever is greater. To obtain the best performance in most cases, set *nservers* to at least sixteen.

AUTHOR

nfsd was developed by Sun Microsystems, Inc.

SEE ALSO

mountd(1M), exports(4).

NAME

nfsstat - Network File System statistics

SYNOPSIS

nfsstat [**-cmnrsz**]

DESCRIPTION

nfsstat displays statistical information about the NFS (Network File System) and RPC (Remote Procedure Call), interfaces to the kernel. It can also be used to reinitialize this information. If no options are given, the default is

nfsstat -cnrs

That is, display everything, but reinitialize nothing.

OPTIONS

-c Display client information. Only the client side NFS and RPC information will be printed. Can be combined with the **-n** and **-r** options to print client NFS or client RPC information only.

-m Display statistics for each NFS mounted file system. This includes the server name and address, mount flags, current read and write sizes, the retransmission count, and the timers used for dynamic retransmission. The **srtt** value contains the smoothed round trip time, the **dev** value contains the estimated deviation, and the **cur** value is the current backed-off retransmission value.

-n Display NFS information. NFS information for both the client and server side will be printed. Can be combined with the **-c** and **-s** options to print client or server NFS information only.

-r Display RPC information.

-s Display server information.

-z Zero (reinitialize) statistics. This option is for use by the super-user only, and can be combined with any of the above options to zero particular sets of statistics after printing them.

DISPLAYS

The server RPC display includes the following fields:

calls	The total number of RPC calls received.
badcalls	The total number of calls rejected by the RPC layer (the sum of **badlen** and **xdrcall** as defined below).
nullrecv	The number of times an RPC call was not available when it was thought to be received.
badlen	The number of RPC calls with a length shorter than a minimum-sized RPC call.
xdrcall	The number of RPC calls whose header could not be XDR decoded.

The server NFS display shows the number of NFS calls received (**calls**) and rejected (**badcalls**), and the counts and percentages for the various calls that were made.

The client RPC display includes the following fields:

calls	The total number of RPC calls made.
badcalls	The total number of calls rejected by the RPC layer.
retrans	The number of times a call had to be retransmitted due to a timeout while waiting for a reply from the server.
badxid	The number of times a reply from a server was received which did not correspond to any outstanding call.
timeout	The number of times a call timed out while waiting for a reply from the server.
wait	The number of times a call had to wait because no client handle was available.
newcred	The number of times authentication information had to be refreshed.
timers	The number of times the calculated time-out value was greater than or equal to the minimum specified time-out value for a call.

The client NFS display shows the number of calls sent and rejected, as well as the number of times a **CLIENT** handle was received (**nclget**), the number of times a call had to sleep while awaiting a handle (**nclsleep**), as well as a count of the various calls and their respective percentages.

AUTHOR

nfsstat was developed by Sun Microsystems, Inc.

NAME

nis_cachemgr - maintains a cache containing location information about NIS+ servers

SYNOPSIS

/usr/sbin/nis_cachemgr [**-i**] [**-n**] [**-v**]

DESCRIPTION

The **nis_cachemgr** daemon maintains a cache of the NIS+ directory objects. The cache contains location information necessary to contact the NIS+ servers that serve the various directories in the name space. This includes transport addresses, information neeeded to authenticate the server, and a time to live field which gives a hint on how long the directory object can be cached. The cache helps to improve the performance of the clients that are traversing the NIS+ name space. **nis_cachemgr** should be running on all the machines that are using NIS+. However, it is not required that the **nis_cachemgr** program be running in order for NIS+ requests to be serviced.

The cache maintained by this program is shared by all the processes that access NIS+ on that machine. The cache is maintained in a file that is memory mapped (see *mmap* (2)) by all the processes. On startup, **nis_cachemgr** initializes the cache from the cold start file (see *nisinit*(1M)) and preserves unexpired entries that already exist in the cache file. Thus, the cache survives machine reboots.

The **nis_cachemgr** program is normally started from a system startup script.

Note: The **nis_cachemgr** program makes NIS+ requests under the NIS+ principal name of the host on which it runs. Before running **nis_cachemgr**, security credentials for the host should be added to the **cred.org_dir** table in the host's domain using *nisaddcred*(1M). Credentials of type DES will be needed if the NIS+ service is operating at security level 2 (see *rpc.nisd*(1M)). See the *WARNINGS* section, below. Additionally, a '**keylogin -r**' needs to be done on the machine.

nisshowcache can be used to look at the cached objects.

Options

-i Force **nis_cachemgr** to ignore the previous cache file and reinitialize the cache from just the cold start file. By default, the cache manager initializes itself from both the cold start file and the old cache file, thereby maintaining the entries in the cache across machine reboots.

-n Run **nis_cachemgr** in an *insecure* mode. By default, before adding a directory object to the shared cache, on the request of another process on the machine, it checks the encrypted signature on the request to make sure that the directory object is a valid one and is sent by an authorized server. In this mode, **nis_cachemgr** adds the directory object to the shared cache without making this check.

-v This flag sets *verbose* mode. In this mode, the **nis_cachemgr** program logs not only errors and warnings, but also additional status messages. The additional messages are logged using *syslog*(3C) with a priority of **LOG_INFO**.

n

DIAGNOSTICS

The **nis_cachemgr** daemon logs error messages and warnings using syslog (see *syslog*(3C)). Error messages are logged to the **DAEMON** facility with a priority of **LOG_ERR**, and warning messages with a priority of **LOG_WARNING**. Additional status messages can be obtained using the **-v** option.

WARNINGS

If the host principal does not have the proper security credentials in the **cred.org_dir** table for its domain, then running this program without the **'-n'** insecure mode option may significantly *degrade* the performance of processes issuing NIS+ requests.

FILES

/var/nis/NIS_SHARED_DIRCACHE	the shared cache file
/var/nis/NIS_COLD_START	the coldstart file
/etc/init.d/rpc	initialization scripts for NIS+

AUTHOR

nis_cachemgr was developed by Sun Microsystems, Inc.

SEE ALSO

keylogin(1), nisaddcred(1M), nisinit(1M), nisshowcache(1M), rpc.nisd(1M), mmap(2), syslog(3C), nisfiles(4).

NAME

nisaddcred - create NIS+ credentials

SYNOPSIS

nisaddcred [**-p** *principal*] [**-P** *nis_principal*] [**-l** *login_password*] *auth_type*
[*domain_name*]

nisaddcred -r [*nis_principal*] [*domain_name*]

DESCRIPTION

The **nisaddcred** command is used to create security credentials for NIS+ principals. NIS+ credentials serve two purposes. The first is to provide authentication information to various services; the second is to map the authentication service name into an NIS+ principal name.

When the **nisaddcred** command is run, these credentials get created and stored in a table named **cred.org_dir** in the default NIS+ domain. If *domain_name* is specified, the entries are stored in the **cred.org_dir** of the specified domain. Note that the credentials of normal users must be stored in the same domain as their passwords.

It is simpler to add credentials using *nisclient*(1M) because it obtains the required information itself. *nispopulate*(1M) can also be used to add credentials for entries in the **hosts** and the **passwd** NIS+ tables.

NIS+ principal names are used in specifying clients that have access rights to NIS+ objects. For more details, refer to the "Principal Names" subsection of the *nis+*(1) manual page. See *nischmod*(1), *nischown*(1), *nis_objects*(3N), and *nis_groups*(3N). Various other services can also implement access control based on these principal names.

The **cred.org_dir** table is organized as follows :

cname	auth_type	auth_name	public_data	private_data
fred.foo.com.	LOCAL	2990	10,102,44	
fred.foo.com.	DES	unix.2990@foo.com	098...819	3b8...ab2

The **cname** column contains a canonical representation of the NIS+ principal name. By convention, this name is the login name of a user or the host name of a machine, followed by a dot (“.”), followed by the fully qualified “home” domain of that principal. For users, the home domain is defined to be the domain where their **DES** credentials are kept. For hosts, their home domain is defined to be the domain name returned by the *domainname*(1) command executed on that host.

There are two types of *auth_type* entries in the **cred.org_dir** table: those with authentication type **LOCAL** and those with authentication type DES. *auth_type*, specified on the command line in upper or lower case, should be either *local* or *des*.

Entries of type LOCAL are used by the NIS+ service to determine the correspondence between fully qualified NIS+ principal names and users identified by UIDs in the domain containing the **cred.org_dir** table. This correspondence is required when associating requests made using the **AUTH_SYS** RPC authentication flavor (see *rpc_clnt_auth*(3N)) to an NIS+ principal name. It is also required for mapping a UID in one domain to its fully qualified NIS+ principal name whose home domain may be elsewhere. The principal’s credentials for any authentication flavor may then be sought for within the **cred.org_dir** table in the principal’s home domain (extracted from the principal name). The same NIS+ principal may have LOCAL credential entries in more than one domain. Only users, and not machines, have LOCAL credentials. In their home domain, users of NIS+ should have both types of credentials.

The *auth_name* associated with the LOCAL type entry is a UID that is valid for the principal in the domain containing the **cred.org_dir** table. This may differ from that in the principal’s home domain. The public information stored in *public_data* for this type contains a list of GIDs for groups in which the user is a member. The GIDs also apply to the domain in which the table resides. There is no private data associated with this type. Neither a UID nor a principal name should appear more than once among the LOCAL entries in any one **cred.org_dir** table.

The DES *auth_type* is used for Secure RPC authentication (see *secure_rpc*(3N)).

The authentication name associated with the DES *auth_type* is a Secure RPC *netname*. A Secure RPC netname has the form **unix.***id@domain*, where *domain* must be the same as the domain of the principal. For principals that are users, the *id* must be the UID of the principal in the principal’s home domain. For principals that are hosts, the *id* is the host’s name. In Secure RPC, processes running under effective UID

n

0 (root) are identified with the host principal. Unlike LOCAL, there cannot be more than one DES credential entry for one NIS+ principal in the NIS+ namespace.

The public information in an entry of authentication type DES is the public key for the principal. The private information in this entry is the private key of the principal encrypted by the principal's network password.

User clients of NIS+ should have credentials of both types in their home domain. In addition, a principal must have a LOCAL entry in the **cred.org_dir** table of each domain from which the principal wishes to make authenticated requests. A client of NIS+ that makes a request from a domain in which it does not have a LOCAL entry will be unable to acquire DES credentials. An NIS+ service running at security level 2 or higher will consider such users unauthenticated and assign them the name *nobody* for determining access rights.

This command can only be run by those NIS+ principals who are authorized to add or delete the entries in the **cred** table.

If credentials are being added for the caller itself, **nisaddcred** automatically performs a keylogin for the caller.

Options

-p *principal* Use the principal name *principal* to fill the *auth_name* field for this entry. For LOCAL credentials, the name supplied with this option should be a string specifying a UID. For DES credentials, the name should be a Secure RPC netname of the form **unix.***id*@*domain*, as described earlier. If the **-p** option is not specified, the *auth_name* field is constructed from the effective UID of the current process and the name of the local domain.

-P *nis_principal*
Use the NIS+ principal name *nis_principal*. This option should be used when creating LOCAL credentials for users whose home domain is different from the local machine's default domain.

Whenever the **-P** option is not specified, **nisaddcred** constructs a principal name for the entry as follows. When it is not creating an entry of type LOCAL, **nisaddcred** calls **nis_local_principal**, which looks for an existing LOCAL entry for the effective UID of the current process in the **cred.org_dir** table and uses the associated principal name for the new entry. When creating an entry of authentication type LOCAL, **nisaddcred** constructs a default NIS+ principal name by taking the login name of the effective UID for its own process and appending to it a dot (".") followed by the local machine's default domain. If the caller is a superuser, the machine name is used instead of the login name.

-l *login_password*
Use the *login_password* specified as the password to encrypt the secret key for the credential entry. This overrides the prompting for a password from the shell. This option is intended for administration scripts only. Prompting guarantees not only that no one can see your password on the command line using *ps*(1), but it also checks to make sure you have not made any mistakes. **NOTE:** *login_password* does not really HAVE to be the user's password, but if it is, it simplifies logging in.

-r [*nis_principal*]
Remove all credentials associated with the principal *nis_principal* from the **cred.org_dir** table. This option can be used when removing a client or user from the system. If *nis_principal* is not specified, the default is to remove credentials for the current *user*. If *domain_name* is not specified, the operation is executed in the default NIS+ domain.

RETURN VALUE

This command returns **0** on success and **1** on failure.

EXAMPLES

Add a LOCAL entry with a UID **2970** for the NIS+ principal name **fredw.some.domain**:

```
nisaddcred -p 2970 -P fredw.some.domain. local
```

Note that credentials are always added in the **cred.org_dir** table in the domain where **nisaddcred** is run, unless *domainname* is specified as the last parameter on the command line. If credentials are being added from the domain server for its clients, then *domainname* should be specified. The caller should have adequate permissions to create entries in the **cred.org_dir** table.

The system administrator can add a DES credential for the same user:

```
nisaddcred -p unix.2970@some.domain \
           -P fredw.some.domain. des
```

Here, **2970** is the UID assigned to the user, **fredw**. **some.domain** comes from the user's home domain, and **fredw** comes from the password file. Note that DES credentials can be added only after the LOCAL credentials have been added.

Note that the secure RPC netname does not end with a dot ("."), while the NIS+ principal name (specified with the **-P** option) does. This command should be executed from a machine in the same domain as the user.

Add a machine's DES credentials in the same domain:

```
nisaddcred -p unix.foo@some.domain \
           -P foo.some.domain. des
```

Note that no LOCAL credentials are needed in this case.

Add a LOCAL entry with the UID of the current user and the NIS+ principal name of **tony.some.other.domain**:

```
nisaddcred -P tony.some.other.domain. local
```

You can list the **cred** entries for a particular principal with *nismatch*(1).

AUTHOR

nisaddcred was developed by Sun Microsystems, Inc.

SEE ALSO

chkey(1), keylogin(1), nis+(1), nischmod(1), nischown(1), nismatch(1), nistbladm(1), nisclient(1M), nispopulate(1M), nis_local_names(3N), rpc_clnt_auth(3N), secure_rpc(3N), nis_objects(3N), nis_groups(3N).

NOTES

The **cred.org_dir** NIS+ table replaces the maps *publickey.byname* and *netid.byname* used in NIS (YP).

NAME

nisaddent - create NIS+ tables from corresponding /etc files or NIS maps

SYNOPSIS

/usr/lib/nis/nisaddent [**-D** *defaults*] [**-Parv**] [**-t** *table*] *type* [*nisdomain*]

/usr/lib/nis/nisaddent [**-D** *defaults*] [**-Paprmv**] **-f** *file* [**-t** *table*] *type* [*nisdomain*]

/usr/lib/nis/nisaddent [**-D** *defaults*] [**-Parmv**] [**-t** *table*] **-y** *ypdomain* [**-Y** *map*] *type* [*nisdomain*]

/usr/lib/nis/nisaddent -d [-AMq] [**-t** *table*] *type* [*nisdomain*]

DESCRIPTION

nisaddent creates entries in NIS+ tables from their corresponding **/etc** files and NIS maps. This operation is customized for each of the standard tables that are used in the administration of HP-UX systems. The *type* argument specifies the type of the data being processed. Legal values for this type are one of **aliases**, **bootparams**, **ethers**, **group**, **hosts**, **netid**, **netmasks**, **networks**, **passwd**, **protocols**, **publickey**, **rpc**, **services**, **shadow**, or **timezone** for the standard tables, or **key-value** for a generic two-column (key, value) table. For a site specific table, which is not of **key-value** type, one can use *nistbladm*(1) to administer it.

The NIS+ tables should have already been created by *nistbladm*(1), *nissetup*(1M), or *nisserver*(1M).

It is easier to use *nispopulate*(1M) instead of **nisaddent** to populate the system tables.

By default, **nisaddent** reads from the standard input and adds this data to the NIS+ table associated with the *type* specified on the command line. An alternate NIS+ table may be specified with the **-t** option. For type **key-value**, a table specification is required.

Note that the data *type* can be different from the table name (**-t**). For example, the automounter tables have **key-value** as the table type.

Although, there is a *shadow* data type, there is no corresponding *shadow* table. Both the shadow and the passwd data are stored in the passwd table itself.

Files may be processed using the **-f** option, and NIS version 2 (YP) maps may be processed using the **-y** option. The merge option is not available when reading data from standard input.

When a *ypdomain* is specified, the **nisaddent** command takes its input from the **dbm** files for the appropriate NIS map (**mail.aliases**, **bootparams**, **ethers.byaddr**, **group.byname**, **hosts.byaddr**, **netid.byname**, **netmasks.byaddr**, **networks.byname**, **passwd.byname**, **protocols.byname**, **publickey.byname**, **rpc.bynumber**, **services.byname**, or **timezone.byname**). An alternate NIS map may be specified with the **-Y** option. For type **key-value**, a map specification is required. The map must be in the **/var/yp/***ypdomain* directory on the local machine. Note that *ypdomain* is case sensitive. *ypxfr*(1M) can be used to get the NIS maps.

If a *nisdomain* is specified, **nisaddent** operates on the NIS+ table in that NIS+ domain; otherwise the default domain is used.

In terms of performance, loading up the tables is fastest when done through the dbm files (**-y**).

Options

-a Add the file or map to the NIS+ table without deleting any existing entries. This option is the default. Note that this mode only propagates additions and modifications, not deletions.

-d Dump the NIS+ table to the standard output in the appropriate format for the given *type*. For tables of type **key-value**, use *niscat*(1) instead. To dump the **cred** table, dump the **publickey** and the **netid** types.

-f *file* Specify that *file* should be used as the source of input (instead of the standard input).

-m Merge the file or map with the NIS+ table. This is the most efficient way to bring an NIS+ table up to date with a file or NIS map when there are only a small number of changes. This option adds entries that are not already in the database, modifies entries that already exist (if changed), and deletes any entries that are not in the source. Use the **-m** option whenever the database is large and replicated, and the map being loaded differs only in a few entries. This option reduces the number of update messages that have to be sent to the replicas. Also see the **-r** option.

-p Process the password field when loading password information from a file. By default, the password field is ignored because it is usually not valid (the actual password appears in a shadow file).

-q Dump tables in "quick" mode. The default method for dumping tables processes each entry individually. For some tables (e.g., hosts), multiple entries must be combined into a single line, so extra requests to the server must be made. In "quick" mode, all of the entries for a table are retrieved in one call to the server, so the table can be dumped more quickly. However, for large tables, there is a chance that the process will run out of virtual memory and the table will not be dumped.

-r Replace the file or map in the existing NIS+ table by first deleting any existing entries, and then add the entries from the source (**/etc** files, or NIS+ maps). This option has the same effect as the **-m** option. The use of this option is *strongly* discouraged due to its adverse impact on performance, unless there are a large number of changes.

-t *table* Specify that *table* should be the NIS+ table for this operation. This should be a relative name as compared to your default domain or the *domainname* if it has been specified.

-v Verbose.

-y *ypdomain*
Use the **dbm** files for the appropriate NIS map, from the NIS domain *ypdomain*, as the source of input. The files are expected to be on the local machine in the **/var/yp/***ypdomain* directory. If the machine is not an NIS server, use *ypxfr*(1M) to get a copy of the **dbm** files for the appropriate map.

-A All data. This option specifies that the data within the table and all of the data in tables in the initial table's concatenation path be returned.

-D *defaults*
This option specifies a different set of defaults to be used during this operation. The *defaults* string is a series of tokens separated by colons. These tokens represent the default values to be used for the generic object properties. All of the legal tokens are described below.

ttl=*time*
This token sets the default time to live for objects that are created by this command. The value *time* is specified in the format as defined by the *nischttl*(1) command. The default is 12 hours.

owner=*ownername*
This token specifies that the NIS+ principal *ownername* should own the created object. The default for this value is the principal who is executing the command.

group=*groupname*
This token specifies that the group *groupname* should be the group owner for the object that is created. The default is **NULL**.

access=*rights*
This token specifies the set of access rights that are to be granted for the given object. The value *rights* is specified in the format as defined by the *nischmod*(1) command. The default is **----rmcdr---r---**.

-M Master server only. This option specifies that lookups should be sent to the master server. This guarantees that the most up-to-date information is seen at the possible expense that the master server may be busy, or that it may be made busy by this operation.

-P Follow concatenation path. This option specifies that lookups should follow the concatenation path of a table if the initial search is unsuccessful.

-Y *map* Use the **dbm** files for *map* as the source of input.

EXAMPLES

Add the contents of **/etc/passwd** to the **passwd.org_dir** table:

```
cat /etc/passwd | nisaddent passwd
```

Add the shadow information (note that the table type here is **shadow**, not **passwd**, even though the actual information is stored in the **passwd** table):

n

```
cat /etc/shadow | nisaddent shadow
```

Replace the **hosts.org_dir** table with the contents of **/etc/hosts** (in verbose mode):

```
nisaddent -rv -f /etc/hosts hosts
```

Merge the **passwd** map from **myypdomain** with the **passwd.org_dir.nisdomain** table (in verbose mode) (the example assumes that the **/var/yp/myypdomain** directory contains the **yppasswd** map.):

```
nisaddent -mv -y myypdomain passwd nisdomain
```

Merge the **auto.master** map from **myypdomain** with the **auto_master.org_dir** table:

```
nisaddent -m -y myypdomain -Y auto.master \
          -t auto_master.org_dir key-value
```

Dump the **hosts.org_dir** table:

```
nisaddent -d hosts
```

EXTERNAL INFLUENCES

Environment Variables

NIS_DEFAULTS This variable contains a default string that will override the NIS+ standard defaults. If the **-D** switch is used, those values will then override both the **NIS_DEFAULTS** variable and the standard defaults.

NIS_PATH If this variable is set, and neither the *nisdomain* nor the *table* is fully qualified, each directory specified in **NIS_PATH** will be searched until the table is found (see *nisdefaults*(1)).

RETURN VALUE

nisaddent returns **0** on success and **1** on failure.

AUTHOR

nisaddent was developed by Sun Microsystems, Inc.

SEE ALSO

niscat(1), nischmod(1), nisdefaults(1), nistbladm(1), nispopulate(1M), nisserver(1M), nissetup(1M), ypxfr(1M), hosts(4), passwd(4).

NAME

nisclient - initialize NIS+ credentials for NIS+ principals

SYNOPSIS

/usr/lib/nis/nisclient -c [**-x**] [**-o**] [**-v**] [**-l** *network_password*]
 [**-d** *NIS+_domain*] *client_name* **. . .**

/usr/lib/nis/nisclient -i [**-x**] [**-v**] **-h** *NIS+_server_host*
 [**-a** *NIS+_server_addr*] [**-d** *NIS+_domain*] [**-S 0**|**2**]

/usr/lib/nis/nisclient -u [**-x**] [**-v**]

/usr/lib/nis/nisclient -r [**-x**]

DESCRIPTION

The **nisclient** shell script can be used to:

- create NIS+ credentials for hosts and users
- initialize NIS+ hosts and users
- restore the network service environment

NIS+ credentials are used to provide authentication information of NIS+ clients to NIS+ service.

Use the first synopsis (**-c**) to create individual NIS+ credentials for hosts or users. You must be logged in as a NIS+ principal in the domain for which you are creating the new credentials. You must also have write permission to the local "cred" table. The *client_name* argument accepts any valid host or user name in the NIS+ domain (for example, the *client_name* must exist in the **hosts** or **passwd** table). **nisclient** verifies each *client_name* against both the **hosts** and **passwd** tables, then adds the proper NIS+ credentials for hosts or users. Note that if you are creating NIS+ credentials outside of your local domain, the host or user must exist in the **hosts** or **passwd** tables in both the local and remote domains.

By default, **nisclient** will not overwrite existing entries in the credential table for the hosts and users specified. To overwrite, use the **-o** option. After the credentials have been created, **nisclient** will print the command that must be executed on the client machine to initialize the host or the user. The **-c** option requires a network password for the client which is used to encrypt the secret key for the client. You can either specify it on the command line with the **-l** option or the script will prompt you for it. You can change this network password later with *nispasswd*(1) or *chkey*(1).

nisclient -c is not intended to be used to create NIS+ credentials for all users and hosts that are defined in the passwd and hosts tables. To define credentials for all users and hosts, use *nispopulate*(1M).

Use the second synopsis (**-i**) to initialize a NIS+ client machine. **-i** The option can be used to convert machines to use NIS+ or to change the machine's domainname. You must be logged in as super-user on the machine that is to become a NIS+ client. Your administrator must have already created the NIS+ credential for this host by using **nisclient -c** or **nispopulate -C**. You will need the network password your administrator created. **nisclient** will prompt you for the network password to decrypt your secret key and then for this machine's root login password to generate a new set of secret/public keys. If the NIS+ credential was created by your administrator using **nisclient -c**, then you can simply use the initialization command that was printed by the **nisclient** script to initialize this host instead of typing it manually.

To initialize an unauthenticated NIS+ client machine, use the **-i** option with the **-S 0**. With these options, the **nisclient -i** option will not ask for any passwords.

During the client initialization process, files that are being modified are backed up as *files*.no_nisplus. The files that are usually modified during a client initialization are: **/etc/rc.config.d/namesvrs**, **/etc/nsswitch.conf**, **/etc/hosts**, and, if it exists, **/var/nis/NIS_COLD_START**. Note that a file will not be saved if a backup file already exists.

The **-i** option does not set up an NIS+ client to resolve hostnames using DNS. Please refer to the DNS documentation for information on setting up DNS. (See *resolver*(4)).

Use the third synopsis (**-u**) to initialize a NIS+ user. You must be logged in as the user on a NIS+ client machine in the domain where your NIS+ credentials have been created. Your administrator should have already created the NIS+ credential for your username using **nisclient -c** or *nispopulate*(1M). You will need the network password your administrator used to create the NIS+ credential for your username. **nisclient** will prompt you for this network password to decrypt your secret key and then for your login password to generate a new set of secret/public keys.

Use the fourth synopsis (**-r**) to restore the network service environment to whatever you were using before **nisclient -i** was executed. You must be logged in as super-user on the machine that is to be restored. The restore will only work if the machine was initialized with **nisclient -i** because it uses the backup files created by the **-i** option.

Reboot the machine after initializing a machine or restoring the network service.

Options

-a *NIS+_server_addr*	Specifies the IP address for the NIS+ server. This option is used *only* with the **-i** option.
-c	Adds **DES** credentials for NIS+ principals.
-d *NIS+_domain*	Specifies the NIS+ domain where the credential should be created when used in conjuction with the **-c** option. It specifies the name for the new NIS+ domain when used in conjuction with the **-i** option. The default is your current domainname.
-h *NIS+_server_host*	Specifies the NIS+ server's hostname. This option is used *only* with the **-i** option.
-i	Initializes an NIS+ client machine.
-l *network_password*	Specifies the network password for the clients. This option is used *only with the* **-c** option. If this option is not specified, the script will prompt you for the network password.
-o	Overwrite existing credential entries. The default is not to overwrite. This is used *only with the* **-c** option.
-r	Restores the network service environment.
-S 0|**2**	Specifies the authentication level for the NIS+ client. Level **0** is for unauthenticated clients and level **2** is for authenticated (**DES**) clients. The default is to set up with level **2** authentication. This is used *only* with the **-i** option. **nisclient** always uses level **2** authentication (**DES**) for both **-c** and **-u** options. There is no need to run **nisclient** with **-u** and **-c** for level **0** authentication.
-u	Initializes an NIS+ user.
-v	Runs the script in verbose mode.
-x	turns the "echo" mode on. The script just prints the commands that it would have executed. Note that the commands are not actually executed. The default is off.

EXAMPLES

To add the **DES** credential for host *hpws* and user *fred* in the local domain:

```
/usr/lib/nis/nisclient -c hpws fred
```

To add the **DES** credential for host *hpws* and user *fred* in domain *xyz.hp.com.*:

```
/usr/lib/nis/nisclient -c -d xyz.hp.com. hpws fred
```

To initialize host *hpws* as an NIS+ client in domain *xyz.hp.com.* where *nisplus_server* is a server for the domain *xyz.hp.com.*:

```
/usr/lib/nis/nisclient -i -h nisplus_server -d xyz.hp.com.
```

The script will prompt you for the IP address of *nisplus_server* if the server is not found in the **/etc/hosts** file. The **-d** option is needed only if your current domain name is different from the new domain name.

To initialize host hpws as an unauthenticated NIS+ client in domain xyz.hp.com. where nisplus_server is a server for the domain xyz.hp.com.:

```
/usr/lib/nis/nisclient -i -S 0 -h nisplus_server -d xyz.hp.com. \
    -a 129.140.44.1
```

To initialize user *fred* as an NIS+ principal, log in as user *fred* on an NIS+ client machine.

```
/usr/lib/nis/nisclient -u
```

FILES

/var/nis/NIS_COLD_START

This file contains a list of servers, their transport addresses, and their Secure

RPC public keys that serve the machines default domain.

/etc/defaultdomain
the system default domainname

/etc/nsswitch.conf
configuration file for the name-service switch

/etc/hosts local host name database

AUTHOR

nisclient was developed by Sun Microsystems, Inc.

SEE ALSO

chkey(1), keylogin(1), nis+(1), nispasswd(1), keyserv(1M), nisaddcred(1M), nisinit(1M), nispopulate(1M), nsswitch.conf(4), resolver(4).

NAME

nisinit - NIS+ client and server initialization utility

SYNOPSIS

nisinit -r

nisinit -p Y|D|N *parent_domain host* ...

nisinit -c -H *host* | **-B** | **-C** *coldstart*

DESCRIPTION

nisinit initializes a machine to be a NIS+ client or an NIS+ root master server. It may be easier to use *nisclient*(1M) or *nisserver*(1M) to accomplish this same task.

Options

-r Initialize the machine to be a NIS+ root server. This option creates the file **/var/nis/root.object** and initializes it to contain information about this machine. It uses the **sysinfo()** system call to retrieve the name of the default domain.

To initialize the machine as an NIS+ root server, it is advisable to use the **-r** option of *nisserver*(1M), instead of using **nisinit -r**.

-p Y | D | N *parent_domain host ...*
This option is used on a root server to initialize a **/var/nis/parent.object** to make this domain a part of the namespace above it. Only root servers can have parent objects. A parent object describes the namespace "above" the NIS+ root. If this is an isolated domain, this option should not be used. The argument to this option tells the command what type of name server is serving the domain above the NIS+ domain. When clients attempt to resolve a name that is outside of the NIS+ namespace, this object is returned with the error **NIS_FOREIGNNS** indicating that a name space boundary has been reached. It is up to the client to continue the name resolution process.

The parameter *parent_domain* is the name of the parent domain in a syntax that is native to that type of domain. The list of host names that follow the domain parameter are the names of hosts that serve the parent domain. If there is more than one server for a parent domain, the first host specified should be the master server for that domain.

Y Specifies that the parent directory is a NIS version 2 domain.

D Specifies that the parent directory is a DNS domain.

N Specifies that the parent directory is another NIS+ domain. This option is useful for connecting a pre-existing NIS+ subtree into the global namespace.

Note that in the current implementation, the NIS+ clients do not take advantage of the **-p** feature. Also, since the parent object is currently not replicated on root replica servers, it is recommended that this option not be used.

-c Initializes the machine to be a NIS+ client. There are three initialization options available: initialize by coldstart, initialize by hostname, and initialize by broadcast. The most secure mechanism is to initialize from a trusted coldstart file. The second option is to initialize using a hostname that you specify as a trusted host. The third method is to initialize by broadcast and it is the least secure method.

-C *coldstart*
Causes the file *coldstart* to be used as a prototype coldstart file when initializing a NIS+ client. This coldstart file can be copied from a machine that is already a client of the NIS+ namespace. For maximum security, an administrator can encrypt and encode (with *uuencode*(1)) the coldstart file and mail it to an administrator bringing up a new machine. The new administrator would then decode (with **uudecode()**), decrypt, and then use this file with the **nisinit** command to initialize the machine as an NIS+ client. If the coldstart file is from another client in the same domain, the **nisinit** command may be safely skipped and the file copied into the **/var/nis** directory as **/var/nis/NIS_COLD_START**.

-H *hostname*
Specifies that the host *hostname* should be contacted as a trusted NIS+ server. The **nisinit** command will iterate over each transport in the **NETPATH** environment variable and attempt to contact *rpcbind*(1M) on that machine. This hostname *must* be reachable from the client without the name service running. For IP networks this means that there must be an entry in

n

/etc/hosts for this host when **nisinit** is invoked.

-B Specifies that the **nisinit** command should use an IP broadcast to locate a NIS+ server on the local subnet. Any machine that is running the NIS+ service may answer. No guarantees are made that the server that answers is a server of the organization's namespace. If this option is used, it is advisable to check with your system administrator that the server and domain served are valid. The binding information can be dumped to the standard output using the *nisshowcache*(1M) command.

Note that **nisinit -c** will just enable navigation of the NIS+ name space from this client. To make NIS+ your name service, modify the file **/etc/nsswitch.conf** to reflect that. See *nsswitch.conf*(4) for more details.

RETURN VALUE

nisinit returns **0** on success and **1** on failure.

EXAMPLES

This example initializes the machine as an NIS+ client using the host *freddy* as a trusted server.

```
nisinit -cH freddy
```

This example sets up a client using a trusted coldstart file.

```
nisinit -cC /tmp/colddata
```

This example sets up a client using an IP broadcast.

```
nisinit -cB
```

This example sets up a root server.

```
nisinit -r
```

EXTERNAL INFLUENCES

Environment Variables

NETPATH This environment variable may be set to the transports to try when contacting the NIS+ server (see *netconfig*(4)). The client library will only attempt to contact the server using connection oriented transports.

FILES

/var/nis/NIS_COLD_START
This file contains a list of servers, their transport addresses, and their Secure RPC public keys that serve the machine's default domain.

/var/nis/*hostname***/root.object**
This file describes the root object of the NIS+ namespace. It is a standard XDR-encoded NIS+ directory object that can be modified by authorized clients using the **nis_modify()** interface.

/var/nis/*hostname***/parent.object**
This file describes the namespace that is logically above the NIS+ namespace. The most common type of parent object is a DNS object. This object contains contact information for a server of that domain.

/etc/hosts Internet host table.

AUTHOR

nisinit was developed by Sun Microsystems, Inc.

SEE ALSO

nis+(1), uuencode(1), nisclient(1M), nisserver(1M), nisshowcache(1M), hosts(4), netconfig(4), nisfiles(4).

NAME

nislog - display the contents of the NIS+ transaction log

SYNOPSIS

/usr/sbin/nislog [**-h** *num* | **-t** *num*] [**-v**] [*directory* ...]

DESCRIPTION

nislog displays the contents of the NIS+ server transaction log on the standard output. This command can be used to track changes in the namespace. The **/var/nis/***hostname***.log** file contains the transaction log maintained by the NIS+ server. *hostname* is the string returned by **uname -n**. When updates occur, they are logged to this file and then propagated to replicas as log transactions. When the log is checkpointed, updates that have been propagated to the replicas are removed.

The **nislog** command can only be run on an NIS+ server by superuser. It displays the log entries for that server only.

If *directory* is not specified, the entire log is searched. Otherwise, only those log entries that correspond to the specified directories are displayed.

Options

-h [*num*] Display *num* transactions from the "head" of the log. If the numeric parameter is omitted, it is assumed to be **1**. If the numeric parameter is **0**, only the log header is displayed.

-t [*num*] Display *num* transactions from the "tail" of the log. If the numeric parameter is omitted, it is assumed to be **1**. If the numeric parameter is **0**, only the log header is displayed.

-v Verbose mode.

FILES

/var/nis/*hostname***.log**
transaction log

n

AUTHOR

nislog was developed by Sun Microsystems, Inc.

SEE ALSO

nis+(1), uname(1), rpc.nisd(1M), nisfiles(4).

NAME

nisping - send ping to NIS+ servers

SYNOPSIS

/usr/lib/nis/nisping [**-uf**] [**-H** *hostname*] [**-r** | *directory*]

/usr/lib/nis/nisping -C [**-a**] [**-H** *hostname*] [*directory*]

DESCRIPTION

In the first *SYNOPSIS* line, the **nisping** command sends a **ping** to all replicas of a NIS+ directory. Once a replica receives a ping, it will check with the master server for the directory to get updates. Prior to pinging the replicas, this command attempts to determine the last update "seen" by a replica and the last update logged by the master. If these two timestamps are the same, the ping is not sent. The **-f** (force) option will override this feature.

Under normal circumstances, NIS+ replica servers get the new information from the master NIS+ server within a short time. Therefore, there should not be any need to use **nisping**.

In the second *SYNOPSIS* line, the **nisping -C** command sends a checkpoint request to the servers. If no *directory* is specified, the home domain, as returned by *nisdefaults*(1), is checkpointed. If all directories, served by a given server, have to be checkpointed, then use the **-a** option.

On receiving a checkpoint request, the servers would commit all the updates for the given *directory* from the table log files to the database files. This command, if sent to the master server, will also send updates to the replicas if they are out of date. This option is needed because the database log files for NIS+ are not automatically checkpointed. **nisping** should be used at frequent intervals (such as once a day) to checkpoint the NIS+ database log files. This command can be added to the *crontab*(1) file. If the database log files are not checkpointed, their sizes will continue to grow.

Options

-a Checkpoint all directories on the server.

-C Send a request to checkpoint, rather than a ping, to each server. The servers schedule to commit all the transactions to stable storage.

-H *hostname* Only the host *hostname* is sent the ping, checked for an update time, or checkpointed.

-f Force a ping, even though the timestamps indicate there is no reason to do so. This option is useful for debugging.

-r This option can be used to update or get status about the root object from the root servers, especially when new root replicas are added or deleted from the list.

If used without **-u** option, **-r** will send a ping request to the servers serving the root domain. When the replicas receive a ping, they will update their root object if needed.

The **-r** option can be used with all other options except with the **-C** option; the root object need not be checkpointed.

-u Display the time of the last update; no servers are sent a ping.

RETURN VALUE

-1	No servers were contacted, or the server specified by the **-H** switch could not be contacted.
0	Success.
1	Some, but not all, servers were successfully contacted.

EXAMPLES

This example pings all replicas of the default domain:

```
nisping
```

Note that this example will not ping the the **org_dir** and **group_dir** subdirectories within this domain.

This example pings the server *example* which is a replica of the *org_dir.foo.com.* directory:

```
nisping -H example org_dir.foo.com.
```

n

This example checkpoints all servers of the *org_dir.bar.com*. directory.

```
nisping -C org_dir.bar.com.
```

EXTERNAL INFLUENCES

Environment Variables

NIS_PATH If this variable is set, and the NIS+ directory name is not fully qualified, each directory specified will be searched until the directory is found.

AUTHOR

nisping was developed by Sun Microsystems, Inc.

SEE ALSO

crontab(1), nisdefaults(1), nislog(1M), nisfiles(4).

NOTES

If the server specified by the **-H** option does not serve the directory, then no ping is sent.

NAME

nispopulate - populate the NIS+ tables in a NIS+ domain

SYNOPSIS

/usr/lib/nis/nispopulate -Y [**-x**] [**-f**] [**-n**] [**-u**] [**-v**] [**-S 0**|**2**] [**-l** *network_passwd*] [**-d** *NIS+_domain*] **-h** *NIS_server_host* [**-a** *NIS_server_addr*] **-y** *NIS_domain* [*table*] ...

/usr/lib/nis/nispopulate -F [**-x**] [**-f**] [**-u**] [**-v**] [**-S 0**|**2**] [**-d** *NIS+_domain*] [**-l** *network_passwd*] [**-p** *directory_path*] [*table*] ...

/usr/lib/nis/nispopulate -C [**-x**] [**-f**] [**-v**] [**-d** *NIS+_domain*] [**-l** *network_passwd*] [*hosts*|*passwd*]

DESCRIPTION

The **nispopulate** shell script can be used to populate NIS+ tables in a specified domain from their corresponding files or NIS maps. **nispopulate** assumes that the tables have been created either through *nisserver*(1M) or *nissetup*(1M).

The table argument accepts standard names that are used in the administration of HP-UX systems and non-standard *key-value* type tables. See *nisaddent*(1M) for more information on *key-value* type tables. If the table argument is not specified, **nispopulate** will automatically populate each of the standard tables. These standard (default) tables are: **auto_master**, **auto_home**, **ethers**, **group**, **hosts**, **networks**, **passwd**, **protocols**, **services**, **rpc**, **netmasks**, **bootparams**, **netgroup**, **aliases** and **shadow**. Note that the **shadow** table is only used when populating from files. The non-standard tables that **nispopulate** accepts are those of *key-value* type. These tables must first be created manually with the *nistbladm*(1) command.

Use the first *SYNOPSIS* (**-Y**) to populate NIS+ tables from NIS maps. **nispopulate** uses *ypxfr*(1M) to transfer the NIS maps from the NIS servers to the **/var/yp/** *NIS_domain* directory on the local machine. Then, it uses these files as the input source. Note that *NIS_domain* is case sensitive. Make sure there is enough disk space for that directory.

Use the second *SYNOPSIS* (**-F**) to populate NIS+ tables from local files. **nispopulate** will use those files that match the table name as input sources in the current working directory or in the specified directory.

Note that when populating the **hosts** and **passwd** tables, **nispopulate** will automatically create the NIS+ credentials for all users and hosts that are defined in the **hosts** and **passwd** tables, respectively. A network passwd is required to create these credentials. This network password is used to encrypt the secret key for the new users and hosts. This password can be specified using the **-l** option or it will use the default password, "nisplus". **nispopulate** will not overwrite any existing credential entries in the credential table. Use *nisclient*(1M) to overwrite the entries in the **cred** table. It creates both LOCAL and DES credentials for users, and only DES credentials for hosts. To disable automatic credential creation, specify the **-S 0** option.

The third *SYNOPSIS* (**-C**) is used to populate NIS+ credential table with level 2 authentication (**DES**) from the passwd and hosts tables of the specified domain. The valid table arguments for this operation are passwd and hosts. If this argument is not specified then it will use both passwd and hosts as the input source.

If **nispopulate** was earlier used with **-S 0** option, then no credentials were added for the hosts or the users. If later the site decides to add credentials for all users and hosts, then this (**-C**) option can be used to add credentials.

Options

-a *NIS_server_addr*
: specifies the IP address for the NIS server. This option is *only* used with the **-Y** option.

-C populate the NIS+ credential table from passwd and hosts tables using **DES** authentication (security level 2).

-d *NIS+_domain.* specifies the NIS+ domain. The default is the local domain.

-F populates NIS+ tables from files.

-f forces the script to populate the NIS+ tables without prompting for confirmation.

-h *NIS_server_host*	specifies the NIS server hostname from where the NIS maps are copied. This is *only* used with the **-Y** option. This host must already exist in either the NIS+ **hosts** table or **/etc/hosts** file. If the hostname is not defined, the script will prompt you for its IP address, or you can use the **-a** option to specify the address manually.
-l *network_passwd*	specifies the network password for populating the NIS+ credential table. This is *only* used when you are populating the **hosts** and **passwd** tables. The default passwd is **nisplus**.
-n	does not overwrite local NIS maps in **/var/yp/***NISdomain* directory if they already exist. The default is to overwrite the existing NIS maps in the local **/var/yp/***NISdomain* directory. This is *only* used with the **-Y** option.
-p *directory_path*	specifies the directory where the files are stored. This is *only* used with the **-F** option. The default is the current working directory.
-S 0|**2**	specifies the authentication level for the NIS+ clients. Level **0** is for unauthenticated clients and no credentials will be created for users and hosts in the specified domain. Level 2 is for authenticated (**DES**) clients and DES credentials will be created for users and hosts in the specified domain. The default is to set up with level **2** authentication (**DES**). There is no need to run **nispopulate** with **-C** for level **0** authentication.
-u	updates the NIS+ tables (ie., adds, deletes, modifies) from either files or NIS maps. This option should be used to bring an NIS+ table up to date when there are only a small number of changes. The default is to add to the NIS+ tables without deleting any existing entries. Also, see the **-n** option for updating NIS+ tables from existing maps in the **/var/yp** directory.
-v	runs the script in verbose mode.
-x	turns the "echo" mode on. The script just prints the commands that it would have executed. Note that the commands are not actually executed. The default is off.
-Y	populate the NIS+ tables from NIS maps.
-y *NIS_domain*	specifies the NIS domain to copy the NIS maps from. This is *only* used with the **-Y** option. The default domainname is the same as the local domainname.

n

EXTERNAL INFLUENCES

TMPDIR

nispopulate normally creates temporary files in the directory **/tmp**. You may specify another directory by setting the environment variable **TMPDIR** to your chosen directory. If **TMPDIR** is not a valid directory, then **nispopulate** will use **/tmp**.

EXAMPLES

To populate all the NIS+ standard tables in the domain *xyz.hp.com.* from NIS maps of the *yp.hp.com* domain as input source where host *yp_host* is a YP server of *yp.hp.com* :

```
/usr/lib/nis/nispopulate -Y -y yp.hp.com -h yp_host -d xyz.hp.com.
```

To update all of the NIS+ standard tables from the same NIS domain and hosts shown above:

```
/usr/lib/nis/nispopulate -Y -u -y yp.hp.com \
     -h yp_host -d xyz.hp.com.
```

To populate the **hosts** table in domain *xyz.hp.com.* from the hosts file in the **/var/nis/files** directory using "somepasswd" as the network password for key encryption:

```
/usr/lib/nis/nispopulate -F -p /var/nis/files -l somepasswd hosts
```

To populate the passwd table in domain xyz.hp.com. from the passwd file in the **/var/nis/files** directory without automatically creating the NIS+ credentials:

```
/usr/lib/nis/nispopulate -F -p /var/nis/files -d xys.hp.com. \
     -S 0 passwd
```

To populate the credential table in domain xyz.hp.com. for all users defined in the passwd table.

```
/usr/lib/nis/nispopulate -C -d xys.hp.com. passwd
```

To create and populate a non-standard key-value type NIS+ table, "private", from the file **/var/nis/files/private**: (nispopulate assumes that the private.org_dir key-value type table has already been created).

```
/usr/bin/nistbladm -D access=og=rmcd,nw=r \
    -c private key=S,nogw= value=,nogw= private.org.dir
/usr/lib/nis/nispopulate -F -p /var/nis/files private
```

FILES

/etc/hosts	local host name database
/var/yp	NIS(YP) domain directory
/var/nis	NIS+ domain directory
/tmp	

AUTHOR

nispopulate was developed by Sun Microsystems, Inc.

SEE ALSO

nis+(1), nistbladm(1), nisaddcred(1M), nisaddent(1M), nisclient(1M), nisserver(1M), nissetup(1M), rpc.nisd(1M), ypxfr(1M).

NAME

nisserver - set up NIS+ servers

SYNOPSIS

/usr/lib/nis/nisserver -r [**-x**] [**-f**] [**-v**] [**-Y**] [**-d** *NIS+_domain*]
[**-g** *NIS+_groupname*] [**-l** *network_passwd*]

/usr/lib/nis/nisserver -M [**-x**] [**-f**] [**-v**] [**-Y**] **-d** *NIS+_domain*
[**-g** *NIS+_groupname*] [**-h** *NIS+_server_host*]

/usr/lib/nis/nisserver -R [**-x**] [**-f**] [**-v**] [**-Y**] [**-d** *NIS+_domain*] [**-h** *NIS+_server_host*]

DESCRIPTION

The **nisserver** shell script can be used to set up a root master, non-root master, and replica NIS+ servers with level 2 security (**DES**).

When setting up a new domain, this script creates the NIS+ directories (including **groups_dir** and **org_dir**) and system table objects for the domain specified. It does not populate the tables. You will need to use *nispopulate*(1M) to populate the tables.

Use the first *SYNOPSIS* (**-r**) to set up a root master server. You must be logged in as super-user on the server machine.

Use the second *SYNOPSIS* (**-M**) to set up a non-root master server for the specified domain. You must be logged in as an NIS+ principal on a NIS+ machine and have create permission to the parent directory of the domain that you are setting up. The new non-root master server machine must already be an NIS+ client (see *nisclient*(1M)) and have the **rpc.nisd** daemon running (see *rpc.nisd*(1M)).

Use the third *SYNOPSIS* (**-R**) to set up a replica server for both root and non-root domains. You must be logged in as an NIS+ principal on an NIS+ machine and have create permission to the parent directory of the domain that you are replicating. The new replica server machine must already be an NIS+ client (see *nisclient*(1M)) and have the **rpc.nisd** daemon running (see *rpc.nisd*(1M)).

Options

-d *NIS+_domain*	specifies the name for the NIS+ domain. The default is your local domain.
-f	forces the NIS+ server setup without prompting for confirmation.
-g *NIS+_groupname*	specifies the NIS+ group name for the new domain. This option is not valid with **-R** option. The default group is **admin**.*domain.*
-h *NIS+_server_host*	specifies the hostname for the NIS+ server. It must be a valid host in the local domain. Use a fully qualified hostname (for example, hostx.xyz.hp.com.) to specify a host outside of your local domain. This option is ONLY used for setting up non-root master or replica servers. The default for non-root master server setup is to use the same list of servers as the parent domain. The default for replica server setup is the local hostname.
-l *network_password*	specifies the network password with which to create the credentials for the root master server. This option is ONLY used for master root server setup (**-r**option). If this option is not specified, the script will prompt you for the login password.
-r	sets up the server as a root master server. Use the **-R** option to set up a root replica server.
-v	runs the script in verbose mode.
-x	turns the "echo" mode on. The script just prints the commands that it would have executed. Note that the commands are not actually executed. The default is off.
-M	sets up the specified host as a master server. Make sure that *rpc.nisd*(1M) is running on the new master server before this command is executed.
-R	sets up the specified host as a replica server. Make sure that *rpc.nisd*(1M) is running on the new replica server.
-Y	sets up an NIS+ server with NIS-compatibility mode. The default is to set up the server without NIS-compatibility mode.

EXAMPLES

To set up a root master server for domain *hp.com.* :

```
root_server# /usr/lib/nis/nisserver -r -d hp.com.
```

For the following examples make sure that the new servers are NIS+ clients and **rpc.nisd** is running on these hosts before executing **nisserver**.

To set up a replica server for domain *hp.com.* on host *hpreplica* :

```
root_server# /usr/lib/nis/nisserver -R -d hp.com. -h hpreplica
```

To set up a non-root master server for domain *xyz.hp.com.* on host *hpxyz* with the NIS+ groupname as *admin-mgr.xyz.hp.com.* :

```
root_server# /usr/lib/nis/nisserver -M -d xyz.hp.com. \
      -h hpxyz -g admin-mgr.xyz.hp.com.
```

To set up a non-root replica server for domain *xyz.hp.com.* on host *hpabc*:

```
hpxyz# /usr/lib/nis/nisserver -R -d xyz.hp.com. -h hpabc
```

AUTHOR

nisserver was developed by Sun Microsystems, Inc.

SEE ALSO

nis+(1), nisgrpadm(1), nismkdir(1), nisaddcred(1M), nisclient(1M), nisinit(1M), nismkdir(1), nispopulate(1M), nissetup(1M), rpc.nisd(1M).

NAME

nissetup - initialize a NIS+ domain

SYNOPSIS

/usr/lib/nis/nissetup [**-Y**] [*domain*]

DESCRIPTION

nissetup is a shell script that sets up a NIS+ domain to serve clients that wish to store system administration information in a domain named *domain*. This domain should already exist prior to executing this command (see *nismkdir*(1) and *nisinit*(1M)).

A NIS+ domain consists of a NIS+ directory and its subdirectories: **org_dir** and **groups_dir**. **org_dir** stores system administration information and **groups_dir** stores information for group access control.

nissetup creates the subdirectories **org_dir** and **groups_dir** in *domain*. Both subdirectories will be replicated on the same servers as the parent domain. After the subdirectories are created, **nissetup** creates the default tables that NIS+ serves. These are **auto_master**, **auto_home**, **bootparams**, **cred**, **ethers**, **group**, **hosts**, **mail_aliases**, **netmasks**, **networks**, **passwd**, **protocols**, **rpc**, **services**, and **timezone**. The **nissetup** script uses the *nistbladm*(1) command to create these tables. The script can be easily customized to add site specific tables that should be created at setup time.

This command is normally executed just once per domain.

Options

-Y Specify that the domain will be served as both a NIS+ domain as well as an NIS domain using the backward compatibility flag. This will set up the domain to be less secure by making all the system tables readable by unauthenticated clients as well.

AUTHOR

nissetup was developed by Sun Microsystems, Inc.

SEE ALSO

nis+(1), nismkdir(1), nistbladm(1), nisaddent(1M), nisinit(1M), nisserver(1M).

NOTES

While this command creates the default tables, it does not initialize them with data. This is accomplished with the *nisaddent*(1M) command.

It is easier to use the *nisserver*(1M) script to create subdirectories and the default tables.

NAME

nisshowcache - NIS+ utility to print out the contents of the shared cache file

SYNOPSIS

/usr/lib/nis/nisshowcache [**-v**]

DESCRIPTION

nisshowcache prints out the contents of the per-machine NIS+ directory cache that is shared by all processes accessing NIS+ on the machine. By default, **nisshowcache** only prints out the directory names in the cache along with the cache header. The shared cache is maintained by *nis_cachemgr*(1M).

Options

-v Verbose mode. Print out the contents of each directory object, including information on the server name and its universal addresses.

DIAGNOSTICS

Error messages are sent to the *syslogd*(1M) daemon.

FILES

/var/nis/NIS_SHARED_DIRCACHE

AUTHOR

nisshowcache was developed by Sun Microsystems, Inc.

SEE ALSO

nis_cachemgr(1M), syslogd(1M), nisfiles(4).

NAME

nisstat - report NIS+ server statistics

SYNOPSIS

/usr/lib/nis/nisstat [**-H** *host*] [*directory*]

DESCRIPTION

The **nisstat** command queries a NIS+ server for various statistics about its operations. These statistics may vary between implementations and from release to release. Not all statistics are available from all servers. Requesting a statistic from a server that does not support that statistic is never fatal, it simply returns 'unknown statistic.'

By default, statistics are fetched from the server(s) of the NIS+ directory for the default domain. If *directory* is specified, servers for that directory are queried.

Supported statistics for this release are as follows:

root server This reports whether the server is a root server.

NIS compat mode This reports whether the server is running in NIS compat mode.

DNS forwarding in NIS mode
This reports whether the server in NIS compat mode will forward host lookup calls to DNS.

security level This reports the security level of this server.

serves directories This lists the directories served by this server.

Operations This statistic returns results in the form:

OP=*opname***:C=***calls***:E=***errors***:T=***micros*

Where *opname* is replaced by the RPC procedure name or operation, *calls* is the number of calls to this procedure that have been made since the server started running. *errors* is the number of errors that have occurred while processing a call, and *micros* is the average time in microseconds to complete the last 16 calls.

Directory Cache This statistic reports the number of calls to the internal directory object cache, the number of hits on that cache, the number of misses, and the hit rate percentage.

Group Cache This statistic reports the number of calls to the internal NIS+ group object cache, the number of hits on that cache, the number of misses, and the hit rate percentage.

Static Storage This statistic reports the number of bytes the server has allocated for its static storage buffers.

Dynamic Storage This statistic reports the amount of heap the server process is currently using.

Uptime This statistic reports the time since the service has been running.

Options

-H *host* Normally all servers for the directory are queried. With this option, only the machine named *host* is queried. If the named machine does not serve the directory, no statistics are returned.

directory If specified, servers for that directory are queried.

EXTERNAL INFLUENCES

Environment Variables

NIS_PATH If this variable is set, and the NIS+ directory name is not fully qualified, each directory specified will be searched until the directory is found (see *nisdefaults*(1)).

AUTHOR

nisstat was developed by Sun Microsystems, Inc.

SEE ALSO

nisdefaults(1).

NAME

nisupdkeys - update the public keys in a NIS+ directory object

SYNOPSIS

/usr/lib/nis/nisupdkeys [**-a** | **-C**] [**-H** *host*] [*directory*]

/usr/lib/nis/nisupdkeys -s [**-a** | **-C**] **-H** *host*

DESCRIPTION

This command updates the public keys in an NIS+ directory object. When the public key for a NIS+ server is changed, the new key must be propagated to all directory objects that reference that server.

nisupdkeys reads a directory object and attempts to get the public key for each server of that directory. These keys are placed in the directory object and the object is then modified to reflect the new keys.

If *directory* is present, the directory object for that directory is updated. Otherwise the directory object for the default domain is updated.

On the other hand, **nisupdkeys -s** gets a list of all the directories served by *host* and updates those directory objects. This assumes that the caller has adequate permission to change all the associated directory objects. The list of directories being served by a given server can also be obtained by *nisstat*(1M).

Before you do this operation, make sure that the new address/public key has been propagated to all replicas.

Options

-a Update the universal addresses of the NIS+ servers in the directory object. Currently, this only works for the TCP/IP family of transports. This option should be used when the IP address of the server is changed. The server's new address is resolved using **gethostbyname()** on this machine. The **/etc/nsswitch.conf** file must point to the correct source for the *hosts* entry for this resolution to work.

-C Specify to clear rather than set the public key. Communication with a server that has no public key does not require the use of secure RPC.

-H *host* Limit key changes only to the server named *host*. If the hostname is not a fully qualified NIS+ name, then it is assumed to be a host in the default domain. If the named host does not serve the directory, no action is taken.

-s Update all the NIS+ directory objects served by the specified server. This assumes that the caller has adequate access rights to change all the associated directory objects. If the NIS+ principal making this call does not have adequate permissions to update the directory objects, those particular updates will fail and the caller will be notified. If the **rpc.nisd** on *host* cannot return the list of servers it serves, the command will print an error message. The caller would then have to invoke **nisupdkeys** multiple times (as in the first *SYNOPSIS*), once per NIS+ directory that it serves.

EXAMPLES

The following example updates the keys for servers of the *foo.bar.* domain.

```
nisupdkeys foo.bar.
```

This example updates the key for host *fred* which serves the *foo.bar.* domain.

```
nisupdkeys -H fred foo.bar.
```

This example clears the public key for host *wilma* in the *foo.bar.* directory.

```
nisupdkeys -CH wilma foo.bar.
```

This example updates the public key in all directory objects that are served by the host *wilma*.

```
nisupdkeys -s -H wilma
```

AUTHOR

nisupdkeys was developed by Sun Microsystems, Inc.

SEE ALSO

chkey(1), niscat(1), nisaddcred(1M), gethostent(3N), nis_objects(3N).

NOTES

The user executing this command must have modify access to the directory object for it to succeed. The existing directory object can be displayed with the *niscat*(1) command using the **-o** option.

This command does not update the directory objects stored in the **NIS_COLD_START** file on the NIS+ clients.

If a server is also the root master server, then **nisupdkeys -s** cannot be used to update the root directory.

NAME

ntpdate - set the date and time via NTP

SYNOPSIS

ntpdate [**-Bbdpqsuv**] [**-a** *key#*] [**-e** *authdelay*] [**-k** *keyfile*]
[**-o** *version*] [**-p** *samples*] [**-t** *timeout*] *server*[...]

DESCRIPTION

ntpdate sets the local date and time by polling those Network Time Protocol (NTP) server(s) given as the server arguments to determine the correct time. It must be run as root on the local host. A number of samples are obtained from each of the servers specified and a subset of the NTP clock filter and selection algorithms are applied to select the best of these. Note that the accuracy and reliability of **ntpdate** depends on the number of servers, the number of polls each time it is run, and the interval between the runs.

ntpdate can be run manually as necessary to set the host clock, or it can be run from the host startup script to set the clock at boot time. This is useful in some cases to set the clock initially before starting the NTP daemon **xntpd**.

It is also possible to run **ntpdate** from a cron script. However, it is important to note that **ntpdate** with contrived cron scripts is no substitute for the NTP daemon, which uses sophisticated algorithms to maximize accuracy and reliability while minimizing resource use. Finally, since **ntpdate** does not discipline the host clock frequency as does **xntpd**, the accuracy using **ntpdate** is limited.

Time adjustments are made by **ntpdate** in one of two ways. If **ntpdate** determines the clock is in error more than 0.5 seconds, it will simply step the time by calling the **clock_settime** (see *clocks*(2)) system routine. If the error is less than 0.5 seconds, it will slew the time by calling the **adjtime** (see *adjtime*(2)) system routine. The latter technique is less disruptive and more accurate when the error is small, and works quite well when **ntpdate** is run by **cron** (see *cron*(1M)) every hour or two.

ntpdate will decline to set the date if an NTP server daemon (e.g., **xntpd**) is running on the same host. When running **ntpdate** on a regular basis from **cron** as an alternative to running a daemon, doing so once every hour or two will result in precise enough timekeeping to avoid stepping the clock.

COMMAND LINE OPTIONS

ntpdate supports the following options:

-a Enable the authentication function and specify the key identifier to be used for authentication. The keys and key identifiers must match in both the client and server key files. The default is to disable the authentication function.

-B Force the time to always be slewed using the **adjtime** system call, even if the measured offset is greater than +-128 ms. The default is to step the time using the **clock_settime** system call if the offset is greater than +-128 ms. Note that, if the offset is much greater than +-128 ms it can take a long time (hours) to slew the clock to the correct value. During this time the host should not be used to synchronize clients.

-b Force the time to be stepped using the **clock_settime** system call, rather than slewed (default) using the **adjtime** system call. This option should be used when called from a startup file at boot time.

-d Enable the debugging mode, in which **ntpdate** will go through all the steps, but not adjust the local clock. Information useful for general debugging will also be printed.

-e *authdelay* Specify the processing delay to perform an authentication function as the value *authdelay*, in seconds and fraction (see *xntpd*(1M) for details). This number is usually small enough to be negligible for most purposes, though specifying a value may improve timekeeping on very slow CPU's.

-k *keyfile* Specify the path for the authentication key file as the string *keyfile*. The default is **/etc/ntp.keys**. This file should be in the format described in **xntpd**.

-o *version* Specify the NTP version for outgoing packets as the integer version, which can be 1 or 2. The default is 3. This allows **ntpdate** to be used with older NTP versions.

-p *samples* Specify the number of samples to be acquired from each server as the integer samples, with values from 1 to 8 inclusive. The default is 4.

-q	Prints the offset measurement, stratum of the server(s) and delay measurement without adjusting the local clock. This is similar to **-d** option which gives a more detailed debugging information.
-s	Divert logging output from the standard output (default) to the system **syslog** (see *syslog*(3C)) facility. This is designed primarily for convenience of **cron** scripts.
-t *timeout*	Specify the maximum waiting time for a server response as the value timeout, in seconds and fraction. The value is rounded to a multiple of 0.2 seconds. The default is 1 second, a value suitable for polling across a LAN.
-u	Direct **ntpdate** to use an unprivileged port for outgoing packets. This is most useful when behind a firewall, that blocks incoming traffic to privileged ports, and you want to synchronise with hosts beyond the firewall. Note that the **-d** option always uses unprivileged ports.
-v	Prints the **NTP** version number and the offset measurement information.

FILES

/etc/ntp.keys	Contains the encryption keys used by **ntpdate**.

SEE ALSO

adjtime(2), clocks(2), cron(1M), syslog(3C), ntpq(1M), xntpd(1M), xntpdc(1M).

DARPA Internet Request For Comments RFC1035 Assigned Numbers.

AUTHOR

ntpdate was developed by Dennis Ferguson at the University of Toronto.

NAME

ntpq - standard Network Time Protocol query program

SYNOPSIS

ntpq [**-dinp**] [**-c** *command*] [*host*] [...]

DESCRIPTION

ntpq is used to query NTP servers, that implement the recommended NTP mode 6 control message format about current state and to request changes in that state. The program may be run either in interactive mode or controlled mode using command line arguments. Requests to read and write arbitrary variables can be assembled, with raw and pretty-printed output options available. **ntpq** can also obtain and print a list of peers in a common format by sending multiple queries to the server.

If one or more request options is included on the command line when **ntpq** is executed, each of the requests will be sent to the NTP servers running on each of the hosts given as command line arguments, or on to *localhost* by default. If no request options are given, **ntpq** will attempt to read commands from the standard input and execute these on the NTP server running on the first host given on the command line, again defaulting to *localhost* when no other host is specified. **ntpq** will prompt for commands if the standard input is a terminal device.

ntpq uses NTP mode 6 packets to communicate with the NTP server, and hence can be used to query any compatible server on the network which permits it. Note that since NTP is a UDP protocol this communication will be somewhat unreliable, especially over large distances in terms of network topology. **ntpq** makes one attempt to retransmit requests, and will time out if the remote host is not heard from within a suitable timeout time.

COMMAND LINE OPTIONS

The command line options supported are described below. Specifying a command line option other than **-i** or **-n** will cause the specified query (queries) to be sent to the indicated host(s) immediately. Otherwise, **ntpq** will attempt to read interactive format commands from the standard input.

-c *command*
: Interactive format command. The command is added to the list of commands to be executed on the specified host(s). Multiple **-c** options may be given.

-d Print debugging information.

-i Force **ntpq** to operate in interactive mode. Prompts will be written to the standard output and commands read from the standard input.

-n Output all host addresses in dotted-quad numeric format rather than converting to the canonical host names.

-p Print a list of the peers known to the server as well as a summary of their state. This is equivalent to the **peers** interactive command.

INTERACTIVE COMMANDS

Interactive format commands consist of a keyword followed by zero to four arguments. Only enough characters of the full keyword to uniquely identify the command needs to be typed. The output of a command is normally sent to the standard output, but optionally the output of individual commands may be sent to a file by appending a **>** followed by a file name, on the command line. A number of interactive format commands are executed entirely within the **ntpq** program itself and do not result in NTP mode 6 requests being sent to a server. These are described below.

? [*command_keyword*]

help [*command_keyword*]
: A **?** or **help** by itself will print a list of all the command keywords known to this version of **ntpq**. A **?** or **help** followed by a command keyword will print function and usage information about the command.

addvars [*variable_name=value*][...]

rmvars [*variable_name=value*][...]

clearvars
: The data carried by NTP mode 6 messages consists of a list of items of the form **variable_name = value**, where the **= value** is ignored, and can be omitted in requests to the server to read variables. **ntpq** maintains an internal list in which data to be

n

included in control messages can be assembled, and sent using the **readlist** and **writel-ist** commands described below.

addvars
: This command allows variables and their optional values to be added to the list. If more than one variable is to be added, the list should be comma-separated and not contain white space.

rmvars This command can be used to remove individual variables from the list.

clearlist
: This command removes all variables from the list.

authenticate [**yes**|**no**]
: Normally **ntpq** does not authenticate requests unless they are write requests. The command **authenticate yes** causes **ntpq** to send authentication with all requests it makes. Authenticated requests causes some servers to handle requests slightly differently, and can occasionally melt the CPU in fuzzballs if you turn authentication on before doing a peer display.

cooked Causes output from query commands to be **cooked**. Variables which are recognized by the server will have their values reformatted for human usage.

debug [**more|less|off**]
: Turns internal query program debugging on and off.

delay *milliseconds*
: Specify a time interval to be added to timestamps included in requests which require authentication. This is used to enable (unreliable) server reconfiguration over long delay network paths or between machines whose clocks are unsynchronized. Actually the server does not now require timestamps in authenticated requests, so this command may be obsolete.

host *hostname*
: Set the host to which future queries will be sent. Hostname may be either a host name or a numeric address.

hostnames [**yes**|**no**]
: If **yes** is specified, host names are printed in information displays. If **no** is specified, numeric addresses are printed instead. The default is **yes**, unless modified using the command line **-n** option.

n

keyid keyid-id
: This command allows the specification of a key number to be used to authenticate configuration requests. This must correspond to a key number the server has been configured to use for this purpose.

ntpversion [**1|2|3**]
: Sets the NTP version number which **ntpq** claims in packets. Defaults to 3, Note that mode 6 control messages (and modes) did not exist in NTP version 1. There appears to be no servers left which demand version 1.

quit Exit **ntpq**.

passwd This command prompts you to type in a password (which will not be echoed) which will be used to authenticate configuration requests. The password must correspond to the key configured for use by the **NTP** server for this purpose if such requests are to be successful.

raw Causes all output from query commands to be printed as received from the remote server. The only formatting/interpretation done on the data is to transform nonascii data into a printable form.

timeout *milliseconds*
: Specify a timeout period for responses to server queries. The default is about 5000 milliseconds. Note that since **ntpq** retries each query once after a timeout, the total waiting time for a timeout will be twice the timeout value set.

CONTROL MESSAGE COMMANDS

Each peer known to an **NTP** server has 16 bit integer association identifier assigned to it. NTP control messages which carry peer variables must identify the peer, the values it corresponds to by including its association ID. An association ID of 0 is special, and indicates the variables are system variables, whose names are drawn from a separate name space.

Control message commands result in one or more NTP mode 6 messages being sent to the server, and cause the data returned to be printed in some format. Most commands currently implemented send a single message and expect a single response. The current exceptions are the **peers** command, which will send a preprogrammed series of messages to obtain the data it needs, and the **mreadlist** and **mreadvar** commands, which will iterate over a range of associations. The supported control messages are listed below:

associations
Obtains and prints a list of association identifiers and peer status for in-spec peers of the server being queried. The list is printed in columns. The first of these columns is an index numbering the associations from 1 for internal use, the second column is the actual association identifier returned by the server and the third column is the status word for the peer. This is followed by a number of columns containing data decoded from the status word. Note that the data returned by the **associations** command is cached internally in **ntpq**. The index is then of use when dealing with stupid servers which use association identifiers which are hard for humans to type, in that for any subsequent commands which require an association identifier as an argument, the form and index may be used as an alternative.

clockvar [*assocID*][*variable_name[=value*[...]][...]]

cv [*assocID*][*variable_name[=value*[...]][...]]
Requests that a list of the server's clock variables be sent. Servers which have a radio clock or other external synchronization will respond positively to this. If the association identifier is omitted or zero the request is for the variables of the **system clock** and will generally get a positive response from all servers with a clock. If the server treats the clocks as pseudo-peers, then more than one clock connected at once, referencing the appropriate peer association ID will show the variables of a particular clock. Omitting the variable list will cause the server to return a default variable display.

lassociations
Obtains and prints a list of association identifiers and peer status for all associations for which the server is maintaining state. This command differs from the **associations** command only for servers which retain state for out-of-spec client associations (i.e., fuzzballs). Such associations are normally omitted from the display when the **associations** command is used, but are included in the output of **lassociations**.

lpassociations
Print data for all associations, including out-of-spec client associations, from the internally cached list of associations. This command differs from **passociations** command only when dealing with fuzzballs.

lpeers
Similar to **peers** command, except a summary of all associations for which the server is maintaining state is printed. This can produce a much longer list of peers from fuzzball servers.

mreadlist *assocID assocID*

mrl *assocID assocID*
Similar to the **readlist** command, except the query is done for each range of (nonzero) association IDs. This range is determined from the association list cached by the most recent associations command.

mreadvar *assocID assocID* [*variable_name*[*=value*][...]]

mrv *assocID assocID* [*variable_name*[*=value*][...]]
Similar to the **readvar** command, except the query is done for each range of (nonzero) association IDs. This range is determined from the association list cached by the most recent associations command.

opeers
An old form of the **peers** command with the reference ID replaced by the local interface address.

passociations
Prints association data concerning in-spec peers from the internally cached list of associations. This command performs identically to the **associations** except that it displays the internally stored data rather than making a new query.

peers Obtains a list of in-spec peers of the server, along with a summary of each peer's state. Summary information includes the address of the remote peer, the reference ID (0.0.0.0 if the refID is unknown), the stratum of the remote peer, the type of the peer (local, unicast, multicast or broadcast), when the last packet was received, the polling interval, in seconds, the reachability register, in octal, and the current estimated delay, offset and dispersion of the peer, all in seconds.

The character in the left margin indicates the fate of this peer in the clock selection process. The codes mean:

`<BLANK>`	discarded due to high stratum and/or failed sanity checks;
`x`	designated falseticker by the intersection algorithm;
`.`	culled from the end of the candidate list;
`-`	discarded by the clustering algorithm;
`+`	included in the final selection set;
`#`	selected for synchronization but distance exceeds maximum;
`*`	selected for synchronization; and
`o`	selected for synchronization, PPS signal in use.

Note that since the **`peers`** command depends on the ability to parse the values in the responses it gets, it may fail to work from time to time with servers which poorly control the data formats. The contents of the host field may be one of four forms. It may be a host name, an IP address, a reference clock implementation name with its parameter or **`REFCLK`**(<*implementation number*>, <*parameter*>). On **`hostnames no`** only IP-addresses will be displayed.

`pstatus` *assocID*
Sends a read status request to the server for the given association. The names and values of the peer variables returned will be printed. Note that the status word from the header is displayed preceding the variables, both in hexadecimal and in English.

`readlist` [*assocID*]

`rl` [*assocID*]
Requests that the values of the variables in the internal variable list be returned by the server. If the association ID is omitted or is 0, the variables are assumed to be system variables. Otherwise they are treated as peer variables. If the internal variable list is empty, a request is sent without data, which should induce the remote server to return a default display.

`readvar` *assocID variable_name* [=*value*][...]]

`rv` *assocID variable_name* [=*value*][...]]
Requests that the values of the specified variables be returned by the server by sending a read variables request. If the association ID is omitted or is given as zero, the variables are system variables. Otherwise they are peer variables and the values returned will be those of the corresponding peer. Omitting the variable list will send a request with no data which should induce the server to return a default display.

`writevar` *assocID variable_name* [=*value*][...]]
Similar to the **`readvar`** command, except the specified variables are written instead of read.

`writelist` [*assocID*]
Similar to the **`readlist`** command, except the internal list variables are written instead of read.

WARNINGS

The **`peers`** command is non-atomic and may occasionally result in spurious error messages about invalid associations occurring and terminating the command. The timeout time is a fixed constant, which means a long wait for timeouts since it assumes a worst case.

FILES

`/etc/ntp.keys` Contains the encryption keys used for authentication.

AUTHOR

`ntpq` was developed by Dennis Ferguson at the University of Toronto.

SEE ALSO

ntpdate(1M), xntpd(1M), xntpdc(1M).

DARPA Internet Request For Comments RFC1035 Assigned Numbers.

n

NAME

ocd - outbound connection daemon used by DDFA software

SYNOPSIS

ocd -f*pseudonym* **-n***node_name* [**-b***board_no*] [**-c***config_file*] [**-l***log_level*] [**-p***port_no*]

DESCRIPTION

The Outbound Connection Daemon (**ocd**) is part of the Data Communications and Terminal Controller (DTC) Device File Access (DDFA) software. It manages the connection and data transfer to the remote terminal server port. It can be spawned from the Dedicated Port Parser (**dpp**) or run directly from the shell.

For performance reasons, **ocd** does not have a debug mode. However, a version called **ocdebug** with debug facilities is available.

See *ddfa*(7) for more information on how to configure the DDFA software and for an explanation of how it works.

ocd logs important messages and error conditions to **/var/adm/syslog**.

Options

ocd recognizes the following options:

-b*board_no* The board number of a DTC. If it is omitted, the port number option must contain the full TCP service port address. The **-b** and **-p** options must not be used if the IP address given in the **-n** option is the IP address of a port.

If the **-n** option explicitly names a terminal server port, the **-b** option is not needed.

-c*config_file* Specify the name (including the absolute path) of the configuration file used to profile the terminal server port. If this option is omitted, the default values specified in the default **pcf** file (**/usr/examples/ddfa/pcf**) are used. If the file specified does not exist, an error message is logged and the following values are used (note that the values for **open_tries** and **open_timer** are different from the default values):

```
telnet_mode:     enable
timing_mark:     enable
telnet_timer:    120
binary_mode:     disable
open_tries:      0
open_timer:      0
close_timer:     0
status_request:  disable
status_timer:    30
eight_bit:       disable
tcp_nodelay:     enable
```

-f*pseudonym* The absolute or relative path to the device file that is linked by the software to the reserved **pty**. Applications use *pseudonym* and not the dynamically allocated **pty** slave.

-l*log_level* Specify the logging level. It determines the severity of messages sent to **/var/adm/syslog**. The logging levels (and how they relate to system logging levels) are as follows:

0	Log only LOG_CRIT messages.
1	Log only LOG_CRIT and LOG_ERR messages.
2	Log only LOG_CRIT, LOG_ERR, and LOG_WARNING messages.
3	Log all messages.

If this option is omitted, the logging level is set to 1.

-n*node_name* The IP address of the terminal server or the port.

-p*port_no* A DTC port number or, if the **-b** option is omitted, the TCP port service address that will be used by the software to access the port. If the value is omitted, the value **23** (Telnet) is used by default.

In order to shutdown every **ocd** running without restarting them, the following command can be executed:

```
kill -15 ‘ps -e | grep ocd | awk ’{print $1}’‘
```

WARNINGS

In order to ensure that commands (such as **ps**) display the correct device file name (that is, the *pseudonym*), all pseudonyms should be placed into the directory **/dev/telnet**. If pseudonyms are not specified for placement in this directory, the correct display of device file names with many commands is not guaranteed.

In addition, in order to ensure that commands (such as **w**, **passwd**, **finger**, and **wall**) work correctly, each pseudonym must be unique in its first 17 characters (including the directory prefix **/dev/telnet/**). If pseudonyms are not unique in their first 17 characters, the correct functioning of many commands is not guaranteed.

Also, in order to reliably handle timing mark negotiations (and ensure that files printing on a printer attached to a terminal server have been completely flushed to that printer), the following line must be added near the end of each printer interface script for printers attached to a terminal server:

```
stty exta <&1 2>/dev/null
```

The printer interface scripts reside in the directory **/etc/lp/interface**. The line must be added just prior to the final **exit** command in each printer interface script.

If this line is not added as specified, the printing reliability of printers attached to a terminal server is not guaranteed.

FILES

/usr/examples/ddfa/dp
/usr/examples/ddfa/pcf
/usr/sbin/dpp
/usr/sbin/ocd
/usr/sbin/ocdebug
/var/adm/dpp_login.bin
/var/adm/utmp.dfa

SEE ALSO

dpp(1M), ocdebug(1M), syslog(3C), dp(4), pcf(4), ddfa(7).

NAME

ocdebug - outbound connection daemon debug utility used by DDFA software

SYNOPSIS

ocdebug -f*pseudonym* **-n***node_name* [**-b***board_no*] [**-c***config_file*] [**-d***debug_level*] [**-l***log_level*] [**-p***port_no*]

DESCRIPTION

The **ocdebug** daemon is the debugging version of the Outbound Connection Daemon (**ocd**). **ocd** is part of the Data Communications and Terminal Controller (DTC) Device File Access (DDFA) software. It manages the connection and data transfer to the remote terminal server port.

See *ddfa*(7) for more information on how to configure the DDFA software and for an explanation of how it works.

Debugging may be toggled interactively by sending the **SIGUSR1** signal to the process using:

kill -16 *pid*.

ocdebug logs important messages and error conditions to **/var/adm/syslog**. Debug messages are logged to the file **/var/adm/ocd***pid* and the file name is displayed at the start of debugging.

Options

ocdebug recognizes the following options. Apart from the **-d** option they are the same as the **ocd** options.

-b*board_no* Specify the board number of a DTC. If it is omitted, the port number option must contain the full TCP service port address. The **-b** and **-p** options must not be used if the IP address given in the **-n** option is the IP address of a port.

If the **-n** option explicitly names a terminal server port, the **-b** option is not needed.

-c*config_file* Specify the name (including the absolute path) of the configuration file used to profile the terminal server port. If this value is omitted, the values specified in the default **pcf** file (**/usr/examples/ddfa/pcf**) are used. If the file specified does not exist, an error message is logged and the following values are used (note that the values for *open_tries* and *open_timer* are different from the default values):

```
telnet_mode:      enable
timing_mark:      enable
telnet_timer:     120
binary_mode:      disable
open_tries:       0
open_timer:       0
close_timer:      0
status_request:   disable
status_timer:     30
eight_bit:        disable
tcp_nodelay:      enable
```

-d*debug_level* Specify the level of debugging. Levels can be added together to accumulate debugging functions. For example, **-d7** enables all levels and **-d3** enables only the first two levels. The levels are:

0	No debug messages.
1	Trace procedure entry/exit logged.
2	Additional tracking messages logged.
4	Data structures dumped.

-f*pseudonym* Specify the absolute or relative path to the device file, which is linked by the software to the reserved **pty**. Applications use the pseudonym and not the dynamically allocated **pty** slave.

-l*log_level* Specify the logging level. It determines the severity of messages sent to **/var/adm/syslog**. The logging levels (and how they relate to system logging levels) are as follows:

0	Log only LOG_CRIT messages.

O

1	Log only LOG_CRIT and LOG_ERR messages.
2	Log only LOG_CRIT, LOG_ERR, and LOG_WARNING messages.
3	Log all messages.

If it is omitted, the logging level is set to 1.

-n*node_name* Specify the IP address of the terminal server or the port.

-p*port_no* Specify a DTC port number or, if the **-b** option is omitted, the TCP port service address that will be used by the software to access the port. If the value is omitted, the value **23** (Telnet) is used by default.

In order to shutdown every **ocd** running without restarting them, the following command can be executed:

```
    kill -15 ‘ps -e | grep ocd | awk ’{print $1}’‘
```

WARNINGS

In order to ensure that commands (such as *ps*) display the correct device file name (that is, the *pseudonym*), all pseudonyms should be placed into the directory **/dev/telnet**. If pseudonyms are not specified for placement in this directory, the correct display of device file names with many commands is not guaranteed.

In addition, in order to ensure that commands (such as **w**, **passwd**, **finger**, and **wall**) work correctly, each pseudonym must be unique in its first 17 characters (including the directory prefix **/dev/telnet/**). If pseudonyms are not unique in their first 17 characters, the correct functioning of many commands is not guaranteed.

Also, in order to reliably handle timing mark negotiations (and ensure that files printing on a printer attached to a terminal server have been completely flushed to that printer), the following line must be added near the end of each printer interface script for printers attached to a terminal server:

```
    stty exta <&1 2>/dev/null
```

The printer interface scripts reside in the directory **/etc/lp/interface**. The line must be added just prior to the final ’exit’ command in each printer interface script.

If this line is not added as specified, the printing reliability of printers attached to a terminal server is not guaranteed.

FILES

/usr/examples/ddfa/dp
/usr/examples/ddfa/pcf
/usr/sbin/dpp
/usr/sbin/ocd
/usr/sbin/ocdebug
/var/adm/dpp_login.bin
/var/adm/ocd*pid*
/var/adm/syslog
/var/adm/utmp.dfa

SEE ALSO

dpp(1M), ocd(1M), syslog(3C), dp(4), pcf(4), ddfa(7).

NAME

opx25 - execute HALGOL programs

SYNOPSIS

/usr/lbin/uucp/X25/opx25 [**-f** *scriptname*] [**-c** *char*] [**-o***file-descriptor*] [**-i***file-descriptor*] [**-n***string*] [**-d**] [**-v**]

DESCRIPTION

HALGOL is a simple language for communicating with devices such as modems and X.25 PADs. It has simple statements similar to **send** *xxx* and **expect** *yyy* that are described below.

Options:

opx25 recognizes the following options:

-f *script*	Causes **opx25** to read script as the input program. If **-f** is not specified, **opx25** reads the standard input as a script.
-c *char*	Causes **opx25** to use *char* as the first character in the input stream instead of actually reading it from the input descriptor. This is useful sometimes when the program that calls **opx25** is forced to read a character but then cannot “unread” it.
-o *number*	Causes **opx25** to use *number* for the output file descriptor (i.e., the device to use for **send**). The default is 1.
-i *number*	Causes **opx25** to use 'number' for the input file descriptor (ie, the device to use for 'expect'). The default is 0.
-n *string*	Causes **opx25** to save this string for use when **\#** is encountered in a **send** command.
-d	Causes **opx25** to turn on debugging mode.
-v	Causes **opx25** to turn on verbose mode.

An **opx25** script file contains lines of the following types:

(empty)	Empty lines are ignored.
/	Lines beginning with a slash (**/**) are ignored (comments)
ID	*ID* denotes a label, and is limited to alphanumerics or _.
send *string*	*string* must be surrounded by double quotes. The text is sent to the device specified by the **-o** option. Non-printable characters are represented as in C; i.e., as \DDD, where DDD is the octal ascii character code. **\#** in a send string is the string that followed the **-n** option.
break	Send a break "character" to the device.
expect *number string*	Here *number* is how many seconds to wait before giving up. 0 means wait forever, but this is not advised. Whenever *string* appears in the input within the time allotted, the command succeeds. Thus, it is not necessary to specify the entire string. For example, if you know that the PAD will send several lines followed by an **@** prompt, you could just use **@** as the string.
run *program args*	The program (**sleep**, **date**, etc.) is run with the args specified. Do not use quotes here. Also, the program is invoked directly (using **execp**), so wild cards, redirection, etc. are not possible.
error *ID*	If the most recent expect or run encountered an error, go to the label *ID*.
exec *program args*	Similar to **run**, but does not fork.
echo *string*	Similar to **send**, but goes to standard error instead of to the device.
set debug	Sets the program in debug mode. It echoes each line to **/tmp/opx25.log**, as well as giving the result of each expect and run. This can be useful for writing new scripts. The command **set nodebug** disables this feature.

O

set log Sends subsequent incoming characters to **/var/uucp/.Log/LOGX25**. This can be used in the ***.in** file as a security measure, because part of the incoming data stream contains the number of the caller. There is a similar feature in **getx25**; it writes the time and the login name into the same logfile. The command **set nolog** disables this feature.

set numlog Similar to **set log**, but better in some cases because it sends only digits to the log file, and not other characters. The command **set nonumlog** disables this feature.

timeout *number*
Sets a global timeout value. Each expect uses time in the timeout reservoir; when this time is gone, the program gives up (exit 1). If this command is not used, there is no global timeout. Also, the global timeout can be reset any time, and a value of 0 turns it off.

exit *number* Exits with this value. 0 is success; anything else is failure.

To perform a rudimentary test of configuration files, run **opx25** by hand, using the **-f** option followed by the name of the script file. **opx25** then sends to standard output and expects from standard input; thus you can type the input, observe the output, and use the **echo** command to see messages. See the file **/usr/lbin/uucp/X25/ventel.out** for a good example of HALGOL programming.

AUTHOR

opx25 was developed by HP.

SEE ALSO

getx25(1), uucp(1).

NAME

ospf_monitor - monitor OSPF (Open Shortest Path First protocol) gateways

SYNOPSIS

ospf_monitor *mon_db_file*

DESCRIPTION

Use the **ospf_monitor** command to query OSPF routers. The **ospf_monitor** command operates in interactive mode. It allows the user to query the various OSPF routers to provide detailed information on IO statistics, error logs, link-state data bases, AS external data bases, the OSPF routing table, configured OSPF interfaces, and OSPF neighbors.

mon_db_file is the complete pathname of a database composed of records configuring destinations for **ospf_monitor** remote commands. Each destination record is a single-line entry which lists the destination IP address, the destination hostname, and an OSPF authentication key (if authentication is activated by the destination). Since authentication keys may be present in the destination records, it is recommended that general access to this database be restricted.

Refer to RFC-1583 (OSPF Specification, version 2) for details about OSPF database and packet formats.

COMMANDS

Upon entering interactive mode, **ospf_monitor** presents this prompt:

[#] dest command params >

From this prompt, you can enter any of the **ospf_monitor** interactive commands. Interactive commands can be interrupted at any time via a keyboard interrupt. Note that the command line length must be less than 200 characters.

Local Commands

? Display all local commands and their functions.

?R Display all remote commands and their functions.

d Display all configured destinations. This command displays *dest_index*, the IP address, and the hostname of all potential **ospf_monitor** command destinations configured in *mon_db_file*.

h Display the command history buffer showing the last 30 interactive commands.

x Exit the **ospf_monitor** program.

@ *remote_command* Send *remote_command* to the same (previous) destination.

@*dest_index remote_command* Send *remote_command* to configured destination *dest_index*.

F *filename* Send all **ospf_monitor** output to *filename*.

S Send all **ospf_monitor** output to stdout.

Remote Commands

a *area_id type ls_id adv_rtr*

Display link state advertisement. *area_id* is the OSPF area for which the query is directed. *adv_rtr* is the router-id of the router which originated this link state advertisement. *type* specifies the type of advertisement to request and should be specified as follows:

1 Request the router links advertisements. They describe the collected states of the router's interfaces. For this type of request, the *ls_id* field should be set to the originating router's Router ID.

2 Request the network links advertisements. They describe the set of routers attached to the network. For this type of request, the *ls_id* field should be set to the IP interface address of the network's Designated Router.

3 Request the summary link advertisements describing routes to networks. They describe inter-area routes, and enable the condensing of routing information at area borders. For this type of request, the *ls_id* field should be set to the destination network's IP address.

4 Request the summary link advertisements describing routes to AS boundary routers. They describe inter-area routes, and enable the condensing of routing information at area borders. For this type of request, the *ls_id* field should be set to the Router ID of the described AS boundary router.

5 Request the AS external link advertisements. They describe routes to destinations external to the Autonomous System. For this type of request, the *ls_id* field should be set to the destination network's IP address.

c Display cumulative log. This log includes input/output statistics for monitor request, hello, data base description, link-state request, link-state update, and link-state ack packets. Area statistics are provided which describe the total number of routing neighbors and number of active OSPF interfaces. Routing table statistics are summarized and reported as the number of intra-area routes, inter-area routes, and AS external data base entries.

e Display cumulative errors. This log reports the various error conditions which can occur between OSPF routing neighbors and shows the number of occurrences for each.

h Display the next hop list. This list of valid next hops is mostly derived from the SPF calculation.

l [*retrans*]
Display the link-state database (except for ASE's). This table describes the routers and networks making up the AS. If *retrans* is non-zero, the retransmit list of neighbors held by this lsdb structure will be printed.

A [*retrans*]
Display the AS external data base entries. This table reports the advertising router, forwarding address, age, length, sequence number, type, and metric for each AS external route. If *retrans* is non-zero, the retransmit list of neighbors held by this lsdb structure will be printed.

o [*which*]
Display the OSPF routing table. This table reports the AS border routes, area border routes, summary AS border routes, networks, summary networks, and AS external networks currently managed via OSPF. If *which* is omitted, all of the above will be listed. If specified, the value of *which* (between 1 and 63) specifies that only certain tables should be displayed. The appropriate value is determined by adding up the values for the desired tables from the following list:

1 Routes to AS border routers in this area.

2 Routes to area border routers for this area.

4 Summary routes to AS border routers in other areas.

8 Routes to networks in this area.

16 Summary routes to networks in other areas.

32 AS routes to non-OSPF networks.

I Display all interfaces. This report shows all interfaces configured for OSPF. Information reported includes the area, interface IP address, interface type, interface state, cost, priority, and the IP address of the DR and BDR for the network.

N Display all OSPF routing neighbors. Information reported includes the area, local interface address, router ID, neighbor IP address, state, and mode.

V Display Gated version information.

AUTHOR
Rob Coltun of University of Maryland

Jeffrey C. Honig of Cornell University

SEE ALSO
gated(1M), gdc(1M), ripquery(1M), gated.conf(4).

GateD Documentation

GateD Configuration Guide

NAME

/usr/sbin/owners - lists owners of outgoing network connections

SYNOPSIS

owners

DESCRIPTION

owners displays a list of established network connections which originate on this system, and indicates the owners of each connection using the **identd** running on this system.

SEE ALSO

sendmail(1M).

NAME

parcreate - create a new partition

SYNOPSIS

parcreate [**-P** *PartitionName*] [**-I** *IPaddress*]
  -c *cell*:[*cellType*]:[*use_on_next_boot*]:[*failure_usage*] [**-c**...]
  [**-b** *path*] [**-t** *path*] [**-s** *path*] [**-r** *cell*] [**-r**...] [**-B**] [**-k** *s_lock*]

DESCRIPTION

The **parcreate** command creates a new partition. This command takes the specified cells (and any attached I/O chassis) from the free cell list and assigns them to the partition.

This command finds an available partition id and assigns it to the new partition. It returns the partition id of the newly created partition.

Options and Arguments

parcreate recognizes the following command line options and arguments:

-P *PartitionName* Specifies the name of the new partition. The characters which can appear in a valid partition name are **a**-**z**, **A**-**Z**, **0**-**9**, **-** (dash), _ (underscore), " " (space) and **.** (period). If the partition name includes space then the name should be enclosed within double quotes.

-I *IPaddress* Specifies the IP address that should be used by management tools (like SAM) to address this partition. This value must be consistent with the IP address that is assigned to the partition once HP-UX is installed and networking is configured.

-c *cell*:[*cellType*]:[*use_on_next_boot*]:[*failure_usage*]
Specifies the cell(s) to be assigned to the partition.

cell specifies the cell id. It can be specified either in the local (*cabinet#*/*slot#*) or global (*slot#*) format. For example, the cell located in cabinet 2, slot 4 is locally identified as 2/4 or globally as simply 20.

cellType specifies the cell type. The current valid *cellType* value is: **base**, which is the default if *cellType* is not specified.

use_on_next_boot specifies whether the cells will participate in a reboot. The current valid values for *use_on_next_boot* are:

y participate in reboot. Default.
n do not participate in reboot.

The valid value for *failure_usage* is:

ri reactivate with interleave. This is the default. Other values will be supported in future releases.

-b *path* Specifies the primary boot path. *path* specifies the physical hardware path.

-t *path* Specifies the alternate boot path. *path* specifies the physical hardware path.

-s *path* Specifies the secondary boot path. *path* specifies the physical hardware path.

-r *cell* Specifies the root alternates. One to four cells can be specified. The first cell specified is the first root alternate cell, the second cell specified is the second root alternate cell, the third cell specified is the third root alternate and the fourth cell specified is the fourth root alternate.

A *cell* can be specified either in the local (*cabinet#*/*slot#*) or global (*slot#*) format. For example, the cell located in cabinet 2, slot 4 is locally identified as 2/4 or globally as simply 20.

-B Specifies to boot the partition. The default is not to boot.

-k *s_lock* Specifies a lock key for the Stable Complex Configuration Data provided by SAM. This option is not available to the command line user.

At least one of the cells specified must contain core I/O with the *use_on_next_boot* flag set to "y".

Root permissions are required to run this command.

EXIT STATUS

The **parcreate** utility exits with one of the following values:

0 Successful completion.

1 Error condition occurred.

EXAMPLES

Create a new partition **myPartition** with 2 cells. One of the cells is located in cabinet 2, slot 4. The *cellType* for the cell is **base**. The *failure_usage* policy for this cell is **ri** - reactivate with interleave.

The other cell is located in cabinet 2, slot 6. The *cellType* is **base** and *failure_usage* policy for this cell is **ri** - reactivate with memory interleave. The *use_on_next_boot* for both cells is **y**.

```
parcreate -P myPartition -c 2/4:base:y:ri -c 2/6:base:y:ri
   -b 0/0/52/2.6 -t 0/0/52/3.6 -s 0/0/52/4.6 -r 2/4
```

Create a new partition **nextPartition** with 2 cells. The cells are located in cabinet 0, slot 4 and slot 6. This example uses the default values for the cells.

```
parcreate -P nextPartition -c 4::: -c 6:::
```

AUTHOR

parcreate was developed by the Hewlett-Packard Company.

SEE ALSO

fruled(1), parstatus(1), partition(1), frupower(1M), parmodify(1M), parremove(1M), parunlock(1M).

NAME

parmgr - partition manager

SYNOPSIS

/usr/sbin/parmgr [[**-t** *task*] [*optional_params*]]

DESCRIPTION

The **parmgr** command provides a single access point to the SuperDome configuration toolset to perform partition administration tasks. Omitting all parameters launches the Partition Manager configuration tool interface.

Partition Manager can also be launched from SAM on SuperDome systems.

parmgr requires superuser (user **root**) privileges to execute successfully.

parmgr does not provide a terminal interface as provided with SAM.

Options

Use one of these **parmgr** *task* options to launch a configuration task:

-h	Display usage instructions.
-t create	Create new partition.
-t modify	Modify an existing partition. Requires a **-p** *par_id* argument. See below.
-t cell_details	Display cell details sheet. Requires a **-c** *cell_num* argument. See below.
-t par_details	Display partition details sheet. Requires a **-p** *par_id* argument. See below.
-t io_details	Display I/O chassis details sheet. Requires **-i** *io_chassis_num* arguments. See below.
-t complex_details	Display SuperDome complex details sheet.

Available *optional_params* are:

-p *par_id*	Partition number.
-c *cell_num*	Cell number. Specify either in the local (cabinet_id/slot_num) or global (cell_ID) format.
-i *io_chassis_num*	The I/O chassis number. Must be specified in the form: cabinet_id/bay_num/chassis_num

EXAMPLES

Run Partition Manager as root:

```
/opt/parmgr/bin/parmgr
```

Create a new partition:

```
parmgr -t create
```

Modify existing partition number 5:

```
parmgr -t modify -p 5
```

Start the partition details sheet for partition 5:

```
parmgr -t par_details -p 5
```

Start the cell details sheet for the cell in cabinet 1, slot 0:

```
parmgr -t cell_details -c 1/0
```

Start the I/O details sheet for cabinet 1, bay 0, chassis 3

```
parmgr -t io_details  -i 1/0/3
```

Start the complex details sheet:

```
parmgr -t complex_details
```

DEPENDENCIES

parmgr runs in an X Window environment.

parmgr requires a minimum of 16 MB of internal memory. More swap space may be required depending on other applications running concurrently.

Partition Manager includes online help that is displayed in a Web browser. In this release of Partition Manager, the online help will only display correctly in the Netscape[tm] web browser, version 4 or later. An appropriate version of Netscape is included in the HP-UX 11i Operating Environment (OE) bundle. For full access to the online help, the OE bundle should be installed on each system running Partition Manager.

AUTHOR

parmgr was developed by HP.

FILES

/opt/parmgr/bin/	**parmgr** executable files.
/opt/webadmin/parmgr/help/$LANG/	**parmgr** online help files.
/opt/parmgr/lib/	**parmgr** internal configuration files.
/usr/var/sam/	**parmgr** uses some SAM executables.
/var/parmgr/	**parmgr** working space, including lock files (if a **parmgr** session dies, it may leave behind a spurious lock file), preferences, logging, and temporary files.
/var/sam/	**parmgr** uses some SAM files.
/var/sam/log/samlog	Unformatted SAM and **parmgr** logging messages. This file should not be modified by users. Use **samlog_viewer** to view the contents of this file (see *samlog_viewer*(1)).
/var/sam/log/samlog.old	Previous SAM and **parmgr** log file created by when **/var/sam/log/samlog** is larger than the user specified limit. Use **samlog_viewer** with its **-f** option to view the contents of this file (see *samlog_viewer*(1)).

SEE ALSO

frupower(1M), frulock(1M), sam(1M), samlog_viewer(1), parstatus(1M), parcreate(1M), parmodify(1M), parunlock(1M), parremove(1M).

These manuals are available on **http://docs.hp.com**:

- *Managing SuperDome Complexes*
- *Installing and Administering Internet Service for HP-UX 11.0*
- *Installing and Administering LAN/9000 Software for HP-UX 11.0*
- *Installing and Administering NFS Services for HP-UX 11.0*
- *X.25/9000 User's Guide*

NAME

parmodify - modify an existing partition

SYNOPSIS

parmodify -p *PartitionNumber*

{ **-a** *cell*:[*cellType*]:[*use_on_next_boot*]:[*failure_usage*] [**-a**...]

| **-m** *cell*:[*cellType*]:[*use_on_next_boot*]:[*failure_usage*] [**-m**...]

| **-I** *IPaddress* | **-r** *cell* [**-r**...] | **-d** *cell* [**-d**...]

| **-b** *path* | **-t** *path* | **-s** *path* | **-P** *PartitionName* | **-B**

| **-k** *s_lock:p_lock* }

DESCRIPTION

The **parmodify** command is used to modify the attributes of an existing partition. This command can modify the following attributes:

- Partition name
- Cell assignment:
  - Add cells to this partition
  - Delete cells from this partition
- Attributes of existing cells:
  - cellType
  - use_on_next_boot
  - failure_usage
- Root alternates
- Primary boot path
- HA Alternate boot path
- Secondary boot path
- Partition's IP address

Options and Arguments

parmodify recognizes the following command line options and arguments:

-p *PartitionNumber*
Specifies the partition to be modified. *PartitionNumber* specifies the unique number (integer) assigned to the partition.

Note: The user has to specify any one or more of the following options.

-a *cell*:[*cellType*]:[*use_on_next_boot*]:[*failure_usage*]
Specifies the cell(s) to be added to the partition.

cell specifies the cell id. It can be specified either in the local (*cabinet#*/*slot#*) or global (*slot#*) format. For example, the cell located in cabinet 2, slot 4 is locally identified as 2/4 or globally as simply 20.

cellType specifies the cell type. The valid *cellType* value is: **base**, which is the default if *cellType* is not specified.

The valid *use_on_next_boot* values are:

y	participate in reboot. Default.
n	do not participate in reboot.

The valid *failure_usage* value for cells is:

ri	reactivate with memory interleave. This is the default. Other values will be supported in future releases.

-m *cell*:[*cellType*]:[*use_on_next_boot*]:[*failure_usage*]
Modify attributes of a cell already assigned to this partition.

cell specifies the cell id. It can be specified either in the local (*cabinet#*/*slot#*) or global (*slot#*) format. For example, the cell located in cabinet 2, slot 4 is locally identified as 2/4 or globally as simply 20.

The valid *cellType* value is: **base**.

The valid *use_on_next_boot* values are:

y participate in reboot.
n do not participate in reboot.

The valid *failure_usage* value for cells is:

ri reactivate with memory interleave. Other values will be supported in future releases.

-I *IPaddress* Specifies the IP address that should be used by management tools (like SAM) to address this partition. This value must be consistent with the IP address that is assigned to the partition once HP-UX is installed and networking is configured.

-r *cell* Specifies the root alternates. One to four cells can be specified. The first cell specified is the first root alternate cell, the second cell specified is the second root alternate cell, the third cell specified is the third root alternate and the fourth cell specified is the fourth root alternate. Use of this option will override previous root alternates selections. So, if the order of a given cell is changing, all of the alternate cells should be specified in the new order.

A *cell* can be specified either in the local (*cabinet#*/*slot#*) or global (*slot#*) format. For example, the cell located in cabinet 2, slot 4 is locally identified as 2/4 or globally as simply 20.

-d *cell* Delete the specified cells from the partition.

-b *path* Specifies the primary boot path. *path* specifies the physical hardware path.

-t *path* Specifies the alternate boot path. *path* specifies the physical hardware path.

-s *path* Specifies the secondary boot path. *path* specifies the physical hardware path.

-P *Partition Name* Specifies the name of the partition. The characters which can appear in a valid partition name are **a**-**z**, **A**-**Z**, **0**-**9**, **-** (dash), _ (underscore), " " (space) and **.** (period). If the partition name includes space then the name should be enclosed within double quotes.

-B Specifies to reboot the partition now. The default is not to boot. If this option is specified and the partition to be modified is not the current partition, then the command proceeds only if the specified partition is not active. Also this option is necessary only if **-a** or **-d** option has been specified. *Note:* The partition in which the command is executing is called the current partition.

-k *s_lock*:*p_lock* Specifies the lock keys provided by SAM for Stable Complex Configuration Data and Partition Configuration Data. The lock keys should always be specified in pairs. If any lock key is not available **-1** should be specified. For example: if the *s_lock* is available but the *p_lock* is not available, then it should be specified as **-k s_lock:-1**.

Note: The **-k** option is not available to the command line user.

Root permissions are required to run **parmodify**.

EXIT STATUS

The **parmodify** utility exits with one of the following values:

0 Successful completion.

1 Error condition occurred.

EXAMPLES

Add a new cell to the existing partition with Partition Number **2**.

```
parmodify -p 2 -a 2/5:base:y:ri
```

Delete a cell from the existing partition with Partition Number **2**.

```
parmodify -p 2 -d 2/5
```

AUTHOR

parmodify was developed by the Hewlett-Packard Company.

p

SEE ALSO

fruled(1), parstatus(1), partition(1), frupower(1M), parcreate(1M), parremove(1M), parunlock(1M).

NAME
parremove - remove an existing partition

SYNOPSIS
parremove -p *PartitionNumber* [**-F**] [**-k** *s_lock*:*p_lock*]

DESCRIPTION
The **parremove** command removes an existing partition. This will unassign all cells from the partition and destroy the partition definition.

Options and Arguments
parremove recognizes the following command line options and arguments:

-p *PartitionNumber*
Specifies the partition to be removed.

PartitionNumber specifies the unique partition number (integer) assigned to the partition.

-F Forcibly remove the partition.

If the partition is inactive, the partition is removed.

If the partition is active and if it is the current partition, the partition is removed.

If the partition is active but is not the current partition, then the partition will not be removed.

Note: The partition in which the command is executing is called the current partition.

-k *s_lock*:*p_lock* Specifies the lock keys provided by SAM for Stable Complex Configuration Data and Partition Configuration Data.

The lock keys should always be specified in pairs. If any lock key is not available, **-1** should be specified. For example: if the *s_lock* is available but the *p_lock* is not available, then it should be specified as **-k s_lock:-1**.

Note: The **-k** option is not available to the command line user.

This command fails if a partition is booted/active. This means all the cells in the partition must be shutdown for reconfiguration before using the **parremove** command. This warning does not apply if the partition being deleted is the current partition and the **-F** option has been specified.

Root permissions are required to run this command.

EXIT STATUS
The **parremove** utility exits with one of the following values:

0 Successful completion.

1 Error condition occurred.

EXAMPLES
Remove the partition whose partition number is **2**.

```
parremove -p 2
```

AUTHOR
parremove was developed by the Hewlett-Packard Company.

SEE ALSO
fruled(1), parstatus(1), partition(1), frupower(1M), parcreate(1M), parmodify(1M), parunlock(1M).

NAME

parunlock - unlock the Stable Complex Configuration Data or Partition Configuration Data

SYNOPSIS

parunlock [**-p** *PartitionNumber*] [**-s**]

parunlock -A

DESCRIPTION

The **parunlock** command unlocks the specified Partition Configuration Data or the Stable Complex Configuration Data.

Options and Arguments

parunlock recognizes the following command line options and arguments:

-p *PartitionNumber*
: Unlock the Partition Configuration Data of the specified partition. *PartitionNumber* specifies the unique partition number (integer) assigned to the partition.

-s
: Unlock the Stable Complex Configuration Data.

-A
: Unlock the Stable Complex Configuration Data and the Partition Configuration Data of all the partitions in the complex.

This command should be used with caution. It should be used only when the system resources are locked due to the abnormal termination of any partition command or other similar applications.

Root permission is required to run this command.

EXIT STATUS

The **parunlock** utility exits with one of the following values:

0 Successful completion.

1 Error condition occurred.

EXAMPLES

Unlock the partition profile of the partition whose partition number is **2**.

```
parunlock -p 2
```

Unlock the Stable Complex Configuration Data.

```
parunlock -s
```

Unlock the Stable Configuration Data and the Partition Configuration Data of all the partitions in the system.

```
parunlock -A
```

AUTHOR

parunlock was developed by the Hewlett-Packard Company.

SEE ALSO

fruled(1), frupower(1M), parcreate(1M), parmodify(1M), parremove(1M), parstatus(1), partition(1).

NAME

pcnfsd - PC-NFS authentication and print request server

SYNOPSIS

/usr/sbin/rpc.pcnfsd

DESCRIPTION

pcnfsd is an RPC server that supports ONC clients on PC (DOS, OS/2, Macintosh, and other) systems. This describes version two of the **pcnfsd** server.

pcnfsd can be started from the **/sbin/init.d/nfs.server** startup script by setting the **PCNFS_SERVER** variable to 1 in **/etc/rc.config.d/nfsconf**, or from the **inetd** daemon (see *inetd*(1M)). It reads the configuration file **/etc/pcnfsd.conf**, if present, and services RPC requests directed to program number 150001. The **pcnfsd** daemon now supports version 1 and version 2 of the PCNFSD protocol.

The requests serviced by **pcnfsd** fall into three categories: authentication, printing, and other. Only the authentication and printing categories have administrative significance.

Authentication

When **pcnfsd** receives a **PCNFSD_AUTH** or **PCNFSD2_AUTH** request, it will "log in" the user by validating the user name and password, returning the corresponding user ID, group IDs, home directory, and umask. It will also append a record to the **wtmp** data base (see *wtmp*(4)). If you do not want PC "logins" recorded in this way, add a line to the **/etc/pcnfsd.conf** file in the form:

wtmp off

By default, **pcnfsd** will only allow authentication or print requests for users with user IDs in the range 101 to 60002 (this corresponds, in SVR4, to the range for nonsystem accounts). To override this, add a line to the **/etc/pcnfsd.conf** file in the form:

uidrange *range* [**,** *range*]...

where each *range* is a user ID number in the form

uid

or an inclusive range of user ID numbers in the form

*uid***-***uid*

NOTE: **pcnfsd** will deny authentication if the **/etc/shells** file is incorrectly setup.

p

Printing

pcnfsd supports a printing model that uses NFS to transfer print data from the client to the server. The client system issues a **PCNFSD_PR_INIT** or **PCNFSD2_PR_INIT** request, and the server returns the path to a spool directory that is exported by NFS for use by the client. **pcnfsd** creates a subdirectory for each client. By default, the parent directory is **/var/spool/pcnfs**, and the name of each subdirectory is the same as its client's host name. To use a different parent directory, add a line to the **/etc/pcnfsd.conf** file in the form:

spooldir *path*

Once a client has mounted the spool directory using NFS, and transferred print data to a file in that directory, it will issue a **PCNFSD_PR_START** or **PCNFSD2_PR_START** request. **pcnfsd** handles most print-related requests by constructing a command based on the printing services of the server's operating system, and executing that command using the identity of the PC user. Because this involves set-user-ID privileges, **pcnfsd** must be run as **root**.

Every print request from a client includes the name of the printer to be used. This name corresponds to a printer that has been configured into the line printer spooling system using the **lpadmin** command.

To process print data in a special way (for example, to print it in landscape mode, or to print it in duplex mode), define a new printer and arrange for the client to print to that printer. There are two ways to define the new printer:

- You can add a new printer to the line printer spooling system that uses a different printer model script, and arrange for the client to use the new printer. Do this using the **lpadmin** command (see *lpadmin*(1M)).

- **pcnfsd** includes a mechanism to define virtual printers known only to **pcnfsd** clients. Each of these printers is defined by an entry in the file **/etc/pcnfsd.conf** using the following format:

  printer *name alias-for command*

  with the following values:

name	The name of the printer, as it will be referred to in print requests from clients.
alias-for	The corresponding name for the printer, as it is defined in the line printer spooling system. For example, a request to display the queue for *name* will be translated into the corresponding request for the printer *alias-for*. If you have defined a printer within **pcnfsd** that has no corresponding printer defined in the line printer spooling system, use a single hyphen (**-**) for this field. For an example, see the definition of the printer **test** in the examples section, below.
command	A command that will be executed whenever a file is printed on *name*. This command is executed by the POSIX shell, **/usr/bin/sh** using the **-c** option. For complex operations, construct an executable shell program and execute that in *command*.

  Within *command* the following tokens will be replaced:

Token	Substitution
$FILE	Replaced by the full path name of the print data file. When the command has been executed, the file will be unlinked.
$USER	Replaced by the user name of the user logged in to the client system.
$HOST	Replaced by the host name of the client system.

Reconfiguration

By checking the modification time (and contents) of the file **/var/spool/lp/pstatus**, **pcnfsd** will detect when printers have been added or deleted, and will rebuild its list of valid printers. However, **pcnfsd** does not monitor the file **/etc/pcnfsd.conf** for updates; if you change this file, you must kill and restart **pcnfsd** for the changes to take effect.

EXAMPLES

Given the following entries for the file **/etc/pcnfsd.conf**:

```
printer abc lj lp -dlj -oraw
printer test - /usr/bin/cp $FILE /usr/tmp/$HOST-$USER
```

If a user on a client system prints a job on printer **abc**, the request will be sent to destination **lj** in raw mode.

If the client requests a list of the print queue for printer **abc**, the **pcnfsd** daemon will translate this into a request for a listing for printer **lj**.

Printer **test** is used only for testing. Any file sent to this printer will be copied into the directory **/usr/tmp**. Any request to list the queue, check the status, etc., of printer **test** will be rejected because *alias-for* has been specified as a hyphen (**-**).

FILES

/etc/pcnfsd.conf
/etc/rc.config.d/nfsconf
/var/spool/lp/pstatus
/var/spool/pcnfs
/etc/shells

SEE ALSO

lp(1), lpstat(1), inetd(1M), lpadmin(1M), wtmp(4).

NAME

pcserver - Basic Serial and HP AdvanceLink server

SYNOPSIS

pcserver [**-n**] [**-l** [*log_file*]] [**-v**]

DESCRIPTION

pcserver is the hostside server program for Basic Serial and AdvanceLink, and is started and terminated by an application program running on a PC.

pcserver supports both the Basic Serial and the AdvanceLink protocols.

Basic Serial offers a library of routines that support a variety of services between a PC and a serially connected host computer, including file transfers and remote interprocess communications.

AdvanceLink is a terminal emulation program that also supports file transfers between a PC and host system over various physical connections.

Options

The following options are recognized by *pcserver*:

-l [*logfile*] This option is now obsolete, but is retained for compatibility with earlier versions of software. Logging is now controlled by the presence or absence of the server.pro file as described in NOTES, below. Enables packet logging and records **pcserver** messages to a specified log file (for debugging). If *logfile* is not specified, the file **s-log** is used in the default logging directory, as defined in the **server.pro** file. **pcserver** looks for a local version of server.pro in the user's home directory. If none is found, it will look for a system-wide version as **/var/adm/server.pro** or **/usr/adm/server.pro**. If the **logfile** exists, logging is appended to it. If the file does not exist, logging is disabled.

-n Informs **pcserver** that a "netmode" for data encryption should be used during special operations (for example, a netmode is needed to mask device control characters when a PAD is being used). The details of the netmode are then negotiated between the **pcserver** and the PC application. For a more comprehensive discussion on netmode, see *Using Basic Serial Connection Files*.

-v Causes **pcserver** to print its version number to standard output and quit.

pcserver is designed to be invoked by a PC application program rather than from the command line. In order for the connection to be correctly established, the PC and host port must be properly configured.

If you are using **pcserver** to manage a session between a PC and a hostside application (via Basic Serial), you may need to use a Basic Serial connection file to actually log in to your account. Establishing connections using Basic Serial connection files is a sensitive operation. Before attempting to use them, you should read the manual *Using Basic Serial Connection Files*.

If you are using **pcserver** to transfer files between a PC and a host machine via Advancelink, use the following AdvanceLink commands:

```
&HOSTCOPY "pcserver"
&TERMINATOR "$"
```

If your prompt does not end with **$**, replace the **$** in the terminator command with the last character in your normal prompt.

To permanently configure AdvanceLink for the HP-UX version of **pcserver**, refer to the *Using AdvanceLink* manual for more information.

NOTES

Packet logging is controlled by the presence or absence of the file **server.pro**

pcserver looks for a local version of **server.pro** in the user's home directory. If none is found, it will look for a system-wide version as **/var/adm/server.pro** or **/usr/adm/server.pro**.

If no logging file is found in these directories, logging is not performed. A commented example of a **server.pro** may be found in **/usr/newconfig/var/adm/server.pro.ex** or **/usr/adm/server.pro.ex**. To make use of this file, copy it to the active file name, **server.pro**, in one of the previously mentioned directory locations.

p

If your screen displays a **Command not found** message when you choose START TRANSFER from AdvLink, either **pcserver** has not yet been installed on your HP-UX system, or it has been installed in a directory that is not part of your current path.

HP-UX treats files containing binary or ASCII data identically. Therefore it is up to the user to specify the desired file type when using **pcserver** to transfer files with Advancelink. The difference between the two is that during ASCII transfers, **pcserver** maps HP-UX line-feed characters to the MS-DOS carriage-return/line-feed pair. This produces incorrect results when transferring a binary file as an ASCII file.

Also, older versions of AdvanceLink show totally inaccurate estimates for file transfer times. This does not interfere with the actual transfer.

If the PC is reset while a transfer is taking place, it may temporarily appear to be a "dead" terminal port. This is no cause for alarm; left to its own devices, **pcserver** will restore the port in a short time. In the worst case, it could take six timeout periods (6×20 = 120 seconds). For faster response, press the Break key a few times to terminate **pcserver** immediately.

FILES

/usr/bin/pcserver	the executable program
/var/adm/server.pro	system-wide logging profile
/usr/adm/server.pro	system-wide logging profile
$HOME/server.pro	local logging profile
/usr/newconfig/var/adm/server.pro.ex	commented inactive example of server.pro
/usr/adm/server.pro.ex	commented inactive example of server.pro

SEE ALSO

Using AdvanceLink	Describes protocol and how to use AdvanceLink.
Using Basic Serial Connection Files	Describes Basic Serial and how connection files should be used.

p

NAME

pdc - processor-dependent code (firmware)

DESCRIPTION

pdc is the firmware that implements all processor-dependent functionality, including initialization and self-test of the processor. Upon completion, it loads and transfers control to the initial system loader (*isl*(1M)). Firmware behavior varies somewhat, depending on the hardware series as described below.

Series 800 Behavior

To load *isl* from an external medium, *pdc* must know the particular device on which *isl* resides. Typically the device is identified by the Primary Boot Path that is maintained by *pdc* in Stable Storage. A *path* specification is a series of decimal numbers each suffixed by '/', indicating bus converters, followed by a series of decimal numbers separated by '.', indicating the various card and slot numbers and addresses. The first number, not specifying a bus converter, is the MID-BUS module number (that is, slot number times four) and followed by the CIO slot number. If the CIO slot contains a terminal card, the next number is the port number, which must be zero for the console.

When the processor is reset after initialization and self-test complete, *pdc* reads the Console Path from Stable Storage, and attempts to initialize the console device. If the initialization fails, *pdc* attempts to find and initialize a console device. Algorithms used to find a console device are model-dependent. *pdc* then announces the Primary Boot, Alternate Boot, and Console Paths.

If *autoboot* (see *isl*(1M)) is enabled, *pdc* provides a 10-second delay, during which time the operator can override the *autoboot* sequence by typing any character on the console. If the operator does not interrupt this process, *pdc* initializes and reads *isl* from the Primary Boot Path. On models that support autosearch, if this path is not valid and *autosearch* (see *isl*(1M)) is enabled, *pdc* then searches through the MID-BUS modules and CIO slots to find a bootable medium. Currently, autosearch is only implemented on the model 825.

If the *autoboot* sequence is unsuccessful, overridden by the operator, or not enabled in the first place, *pdc* interactively prompts the operator for the Boot Path to use. Any required path components that are not supplied default to zero.

The Primary Boot, Alternate Boot, and Console Paths as well as *autoboot* and *autosearch* enable can be modified via *isl*.

Series 700 Behavior

To load *isl* from an external medium, *pdc* must know the particular device on which *isl* resides. Typically the device is identified by the Primary Boot Path that is maintained by *pdc* in Stable Storage. A *path* specification is an I/O subsystem mnemonic that varies according to hardware model.

When the processor is reset after initialization and self-test complete, *pdc* reads the Console Path from Stable Storage, and attempts to initialize the console device. If the initialization fails, *pdc* attempts to find and initialize a console device. Algorithms used to find a console device vary according to hardware model.

If *autoboot* and *autosearch* (see *isl*(1M)) are enabled, *pdc* waits for approximately 10 seconds during which time the operator can override the *autoboot* sequence pressing and holding the ESC (escape) key on the console.

The system then begins a search for potentially bootable devices. If allowed to complete, a list of potentially bootable devices is displayed, labeled with abbreviated path identifiers (P0, P1, etc). A simple menu is then displayed where the user can:

- Boot a specific device, using the abbreviated path identifier, or the full mnemonic.
- Start a device search where the contents are searched for IPL images (note the first search only identified devices and did not check the contents).
- Enter the boot administration level.
- Exit the menu and return to autobooting
- Get help on choices

The search of potentially bootable devices can be aborted by pressing and holding the escape key. The search for device contents can also be aborted by pressing and holding the escape key.

If the operator does not interrupt the search process, *pdc* initializes and reads *isl* from the Primary Boot Path.

If the *autoboot* sequence is unsuccessful, overridden by the operator, or not enabled in the first place, *pdc* executes the device search and enters the menu described above.

The Primary Boot, Alternate Boot, and Console Paths as well as *autoboot* and *autosearch* enable can be modified via *isl* or at the pdc boot administration level.

SEE ALSO

boot(1M), isl(1M).

(TO BE OBSOLETED)

NAME

pddcesetup - configure DCE for the HPDPS

SYNOPSIS

pddcesetup [**force**]

DESCRIPTION

The **pddcesetup** command is used to configure DCE information for the HP Distributed Print Service (HPDPS).

pddcesetup must be run on each HP-UX host in your DCE cell that will execute the HPDPS in the Extended Environment. If a host will run the HPDPS in the Basic Environment only, not in the Extended Environment, then execution of **pddcesetup** is not needed on that host. If you do not intend to execute the HPDPS in a DCE cell, or do not wish to use DCE services in conjunction with the HPDPS, then each host must execute the HPDPS in the Basic Environment, and execution of **pddcesetup** is not needed on any of the hosts in your network. **pddcesetup** must be executed once on each host in your DCE cell that will execute an HPDPS client daemon, spooler, or supervisor in the Extended Environment. **pddcesetup** must be executed before starting any of these HPDPS components on that host.

The first time that **pddcesetup** executes in your DCE cell, it will prompt for and create various DCE security identities and CDS namespace entries that are then configured for the entire cell. On subsequent executions of **pddcesetup**, only local information needed by the local host is configured; the DCE security identities and CDS namespace entries have already been created for the cell.

If the parameter **force** is given, then **pddcesetup** will give you the option to fully recreate security identities and CDS directories. This option is useful if DCE has been only partially configured, or if configuration is accidentally removed after creation. If the **force** parameter is not given, then **pddcesetup** quickly checks to see if it appears the DCE information has already been configured, and if so, **pddcesetup** does not attempt to configure any of this information.

The host on which **pddcesetup** is executed must already be configured as a DCE client or server, using DCE command-line utilities or the SAM program. You must also be DCE logged in using an account with sufficient administrator privileges to modify DCE security and namespace information. The DCE login of the cell administrator for your cell is normally used for this purpose (the default DCE login name is **cell_admin**).

The HPDPS DCE information created by **pddcesetup** is:

- Accounts adm_user and pd_server.
- Principals adm_user and pd_server.
- Groups pd_admin and pd_operator.
- CDS namespace directories and links.
- Initial Access Control List entries.
- Local key table entries for principal pd_server.

EXAMPLES

An example execution follows. This example illustrates the execution of pddcesetup for the first time in a DCE cell.

```
# pddcesetup

Checking whether your host is configured in a DCE cell.

Verifying your DCE login.
You are DCE logged in as <your login>.

Checking whether HPDPS security identities have already been configured
in your cell.

DCE security identities needed for the HPDPS have not yet been
configured in your DCE cell.  The security identities that must be
configured are:

 Accounts   adm_user and pd_server
 Principals adm_user and pd_server
 Groups     pd_admin and pd_operator
```

p

```
Are you ready to configure these identities now (y/n)? y

The new groups and accounts about to be created must be members of a DCE
"organization".  You may use an existing organization if you have
already defined one.

Do you wish to create a new DCE organization (y/n)? y
Please enter the name of the new organization: <organization name>

Creating organization <organization name>.

Creating group pd_admin.

Creating group pd_operator.

Creating principal pd_server.

Creating principal adm_user.

Adding new principals to groups and organizations.

pddcesetup is ready to create DCE accounts pd_server and adm_user.
To accomplish this, you must enter the password for the DCE account under
which you are currently logged in.  If not entered correctly, an attempt
to create the new accounts will generate the error message "data
integrity error".

Please enter the password for your current DCE login account: <password>

Please choose a unique password for new account adm_user.
This account will be used by HPDPS administrators.

Please enter the password for account adm_user: <password>
Please re-enter the password for account adm_user: <password>

Creating account adm_user.

Please choose a unique password for the pd_server account.
This password is used by the HPDPS client daemon, spooler, and
supervisor to automatically DCE login to account pd_server.

Please enter the password for account pd_server: <password>
Please re-enter the password for account pd_server: <password>

Creating account pd_server.

Creating HPDPS directories and links in the DCE CDS namespace.

Creating initial HPDPS Access Control List entries.

Adding entry for pd_server to the local key table.

pddcesetup: DCE setup is complete.
```

p

SEE ALSO

dce_config(1M), dce_login(1M).

HP Distributed Print Service Administration Guide

NAME

pdfck - compare Product Description File to File System

SYNOPSIS

pdfck [**-n**] [**-r** *alternate_root*] *PDF*

DESCRIPTION

pdfck is a program that compares the file descriptions in a PDF (Product Description File) to the actual files on the file system. It is intended as a tool to audit the file system and detect corruption and/or tampering. Differences found are reported in the format described in the *pdfdiff*(1M) manual entry. (Size growth (**-p** option) is not reported.) For a detailed explanation of the PDF fields see *pdf*(4). The command

```
pdfck -r /pseudoroot /system/AL_CORE/pdf
```

is roughly equivalent to

```
mkpdf -r /pseudoroot /system/AL_CORE/pdf - | \
pdfdiff /system/AL_CORE/pdf -
```

Options

pdfck recognizes the following options:

-n Compare numerical representation of user id *uid* and group id *gid* of each file, instead of the usual text representation. If owner or group is recorded in the PDF as a name, look the name up in the **/etc/passwd** or **/etc/group** file, respectively, to find the id number.

-r *alternate_root* *alternate_root* is a string that is prefixed to each pathname in the prototype when the filesystem is being searched for that file. Default is NULL.

EXAMPLES

The following output indicates tampering with **/usr/bin/cat**:

```
/usr/bin/cat: mode(-r-xr-xr-x -> -r-sr-xr-x)(became suid), size(27724 -> 10345),
      checksum(1665 -> 398)
```

WARNING

Use of PDFs is discouraged since this functionality is obsolete and is being replaced with Software Distributor (see *sd*(4)).

FILES

/system/*fileset_name***/pdf** Product Description File of fileset called *fileset_name*.

SEE ALSO

mkpdf(1M), pdfdiff(1M), pdf(4).

NAME

pdfdiff - compare two Product Description Files

SYNOPSIS

pdfdiff [**-n**] [**-p** *percent*] *pdf1 pdf2*

DESCRIPTION

pdfdiff is a program that compares two PDFs (Product Description Files). The PDFs can be generated using the **mkpdf** command (see *mkpdf*(1M)). Individual fields in the PDFs are compared, and differences found in these fields are reported. For a detailed explanation of the PDF fields see *pdf*(4).

The report format is:

> *pathname*: *diff_field*[(*details*)][,...]

diff_field is one of the field names specified in *pdf*(4). The format of *details* is "*oldvalue* –> *newvalue*" and may include an additional "(*added description*)".

A summary of total product growth in bytes, **DEV_BSIZE** disk blocks, and the percentage change in disk blocks is reported. This summary includes growth of all files, including those for which growth did *not* exceed the threshhold *percent*. Format of the growth summary is:

> Growth: *x* bytes, *y* blocks (*z*%)

Options

pdfdiff recognizes the following options:

-n	Compare numerical representation of user ID *uid* and group ID *gid* of each file, instead of the usual text representation. If owner or group is recorded in the PDF as a name, look the name up in the **/etc/passwd** or **/etc/group** file, respectively, to find the ID number.
-p *percent*	specifies a threshhold percentage for file growth. Files having a net size change greater than or equal to this percentage are reported. A decrease in size is reported as a negative number. If **-p** is not specified, a default value of zero percent is used.

EXAMPLES

The following output results when the **/usr/bin/cat** entry in the example from *pdf(4)* is different in the compared PDF:

```
/usr/bin/cat: mode(-r-xr-xr-x -> -r-sr-xr-x)(became suid), size(27724 -> 10345),
      checksum(1665 -> 398)
Growth: -17379 bytes, -17 blocks (-4%)
```

WARNING

Use of PDFs is discouraged since this functionality is obsolete and is being replaced with Software Distributor (see *sd*(4)).

FILES

/system/*fileset_name***/pdf**

SEE ALSO

mkpdf(1M), pdfck(1M), pdf(4).

(TO BE OBSOLETED)

NAME

pdgwcfg - configures HPDPS gateway printers in a Basic environment

SYNOPSIS

pdgwcfg [-a|-m] [-h] [-p] [-v]

DESCRIPTION

The **pdgwcfg** utility simplifies the configuration of HPDPS gateway printers in a Basic (non-DCE Extended) environment by reading an administrator-supplied configuration file **/etc/pdgwcfg.conf** (see *pdgwcfg.conf*(4)). It creates-enables and/or deletes gateway printers as appropriate. Gateway printers are similar to "remote printers" provided by the LP spooler, allowing access to a printer in a foreign (DCE Extended or Basic) environment.

You must have super-user privileges to invoke the utility. The default behavior of the utility will not modify any previously-created gateway printers listed in **/etc/pdgwcfg.conf**. Any new entries will be created and any gateway printers not listed will be deleted.

All output is sent to **$PDBASE/pdgwcfg/error.log** (or **/var/opt/pd/pdgwcfg/error.log** by default). Previous error logs are retained in separate files in this directory for reference and, if used extensively, the directory may need to be cleaned-up periodically. If a severe error is encountered that causes a premature abort, an error message is also sent to stderr.

Options

pdgwcfg uses the following options:

-a Retain all previously-created gateway printers, even if they are no longer listed in **/etc/pdgwcfg.conf**. No gateway printers will be deleted. The administrator must manually remove any unwanted gateway printers.

-m Retain any manually-created gateway printers. These would not contain the text **PDGWCFG-MARKER** in the *descriptor* attribute which is placed there by the **pdgwcfg** utility when it creates a gateway printer. This option is intended for scenarios where **/etc/pdgwcfg.conf** is not used exclusively for gateway printer configuration (e.g. a local sysadmin also creates gateway printers without the utility).

-h Provides invocation syntax help.

-p Preview mode. No changes to the gateway configuration will actually be made. For best results, the local HPDPS system should be running so that the utility can query the system to determine which gateway printers, if any, would be created/deleted.

-v Verbose mode. Provides more extensive output in the **error.log**.

RETURN VALUE

pdgwcfg exits with one of the following values:

0 Successful completion.
1 Failure.

EXTERNAL INFLUENCES

Environment Variables

PATH needs to include at a minimum **/usr/bin:/opt/pd/bin**.

PDBASE affects the location of the **error.log**

EXAMPLES

If it is desirable to have a single configuration file distributed across multiple systems, there are various ways to distribute the configuration file and have the utility invoked to configure a system (e.g. *swinstall*(1M), *rdist*(1), etc.). Each method should be weighed against the administration and security concerns of your particular environment.

The below is just an example using *rdist*(1) and is not intended to be a recommendation.

```
# sample distfile for use with rdist
# copies /etc/pdgwcfg.conf and invokes the pdgwcfg utility
# invoke as ’rdist -b -h -f distfile’
HOSTS = ( host7 host8 )
```

p

```
FILES = ( /etc/pdgwcfg.conf )
${FILES} -> ${HOSTS}
        install ;
        special /etc/pdgwcfg.conf \
     " PATH=/usr/bin:/opt/pd/bin;pdgwcfg" ;
```

To update the configuration on host8 only, one would invoke:

```
rdist -b -h -f distfile -m host8
```

WARNINGS

By default, any gateway printer not listed in **/etc/pdgwcfg.conf** will be removed. **pdgwcfg** does not check for entry modifications in **/etc/pdgwcfg.conf**. See *pdgwcfg.conf*(4) for possible ways to accomplish modifications.

If the descriptor attribute is modified, then the **-m** option will not consider these gateway printers as candidates for deletion because it overrides the **PDGWCFG-MARKER** marker that **pdgwcfg** would have placed in that attribute.

AUTHOR

pdgwcfg was developed by HP.

SEE ALSO

pdgwcfg.conf(4), pdcreate(1), and the *HP Distributed Print Service Administration Guide* (re: gateway printers).

(TO BE OBSOLETED)

NAME

pdstartclient - start the HPDPS client daemon

SYNOPSIS

pdstartclient [**-l** *locale*] [**-p** *port*] [**-q**]

DESCRIPTION

The **pdstartclient** utility is issued by an administrator to start the HPDPS client daemon.

Options

The **pdstartclient** utility uses the following flag:

-l *locale*	Allows you to specify the locale for HPDPS messages in a specific language.
-p *port*	Allows you to specify the port number when starting a HPDPS client in a locale other than the default locale. The port number you assign must not conflict with port numbers in use by other processes. The file **/etc/services** lists the port numbers reserved by other processes.
-q	Allows you to query the status whether the daemon is running or not running without starting the daemon.

EXAMPLES

Start a Daemon

To start the daemon, enter the following:

```
pdstartclient
```

Start a Daemon in a Different Locale

To start the daemon in a Japanese locale and assign port number 1411, enter the following:

```
pdstartclient -l ja_JP.SJIS -p 1411
```

Query the Status of a Daemon

To query the status of a daemon, enter the following:

```
pdstartclient -q; echo $?
```

If the daemon is running, you will receive the following message:

```
The HPDPS daemon is already running
0
```

If the daemon is not running, you will receive a **1**.

To query the status of a daemon running in locale **ja_JP.SJIS**, enter the following:

```
pdstartclient -q -l ja_JP.SJIS
```

To query the status of a daemon running on port 1411, enter the following:

```
pdstartclient -q -p 1411
```

SEE ALSO

pdstopd(1M), pdstartspl(1M), pdstartsuv(1M).

(TO BE OBSOLETED)

NAME

pdstartspl - create or restart an HPDPS spooler

SYNOPSIS

pdstartspl [**-F**] *ServerName*

DESCRIPTION

The **pdstartspl** utility is issued by an administrator to create or restart a spooler. A spooler represents the server that manages the validation, routing, and scheduling of jobs. A spooler contains logical printers and queues.

You can restart a spooler after it is terminated by issuing this same utility.

When you create or restart a spooler, you must specify its name.

Options

The **pdstartspl** utility uses the following flag:

-F Bypasses (does not display) prompts; force creation of a new spooler

Arguments

The argument value identifies the specific object to which the utility applies.

The valid argument value for the **pdstartspl** utility is:

ServerName
Assigns a name to a new spooler or specifies the name of the spooler to restart.

EXAMPLES

Create or Restart a Spooler

To create or restart a spooler, **spool1**, enter the command:

```
pdstartspl spool1
```

SEE ALSO

pdstartclient(1M), pdstartsuv(1M), pdshutdown(1).

(TO BE OBSOLETED)

NAME

pdstartsuv - create or restart an HPDPS supervisor

SYNOPSIS

pdstartsuv [**-F**] *ServerName*

DESCRIPTION

The **pdstartsuv** utility is issued by an administrator to create or restart a supervisor. A supervisor receives jobs from a spooler and manages the printing process. A single supervisor may contain many physical printers. The supervisor may be started on a different HP-UX processor from the spooler, but it must be able to communicate with its printer devices.

If the supervisor already exists but is shut down, the **pdstartsuv** utility restarts it.

Options

The **pdstartsuv** utility uses the following flag:

-F Bypasses (does not display) prompts; force creation of a new supervisor.

Arguments

The argument value identifies the specific object to which the utility applies.

The valid argument value for the **pdstartsuv** utility is:

ServerName
Assigns a name to a new supervisor or specifies the name of the supervisor to restart.

EXAMPLES

Create or Restart a Supervisor

To create or restart a supervisor, **super1**, enter the command:

```
pdstartsuv super1
```

SEE ALSO

pdstartclient(1M), pdstartspl(1M), pdshutdown(1).

(TO BE OBSOLETED)

NAME

pdstopd - stop the HPDPS client daemon

SYNOPSIS

pdstopd

DESCRIPTION

The **pdstopd** utility is issued by an administrator to stop the HPDPS client daemon.

EXAMPLES

Stopping a Daemon

To stop the daemon, enter the command:

```
pdstopd
```

To stop the daemon running in a specific locale, such as **ja_JP.SJIS** enter the following:

```
export LC_ALL=ja_JP.SJIS
pdstopd
export LC_ALL=
```

SEE ALSO

pdstartclient(1M), pdstartspl(1M), pdstartsuv(1M).

NAME

pfs_exportfs - export and unexport directories to PFS clients

SYNOPSIS

/usr/sbin/pfs_exportfs [**-a -u -v**] [*pathname*]

DESCRIPTION

pfs_exportfs makes a local directory or filename available for mounting over the network by PFS clients. It is recommended that a command to invoke **pfs_exportfs** at boot time be added to *rc*(1M). **pfs_exportfs** uses information contained in the **/etc/pfs_exports** file to export *pathname* (which must be specified as a full pathname). The superuser can run **pfs_exportfs** at any time to alter the list or characteristics of exported directories and filenames. Directories and files that are currently exported are listed in the file **/etc/pfs_xtab**.

With no options or arguments, **pfs_exportfs** prints out the list of directories and filenames currently exported.

Options

-a All. Export all pathnames listed in **/etc/pfs_exports**, or if **-u** is specified, unexport all of the currently exported pathnames.

-u Unexport the indicated pathnames.

-v Verbose. Print each directory or filename as it is exported or unexported.

AUTHOR

psf_exportfs was developed by Young Minds, Inc.

FILES

/etc/pfs_exports	static export information
/etc/pfs_xtab	current state of exported pathnames

SEE ALSO

pfs_exports(5).

NAME

pfs_mount, pfs_umount - mount and unmount CD-ROM file systems

SYNOPSIS

pfs_mount [**-v -a**]

pfs_mount [**-v -a -f -n**] [**-t** *type*] [**-x** *xlat*] [**-o** *options*] *filesystem directory*

pfs_mount [**-v -a -f -n**] [**-x** *xlat*] [**-o** *options*] *filesystem* | *directory*

pfs_umount [**-v -a -c**] *filesystem* | *directory*

DESCRIPTION

pfs_mount attaches a named *filesystem* to the file system hierarchy at the pathname location *directory*, which must already exist. If *directory* has any contents prior to the **pfs_mount** operation, these remain hidden until the *filesystem* is once again unmounted. If *filesystem* is of the form *host*:*pathname*, it is assumed to be a remote file system.

In the case of a local mount, **pfs_mount** probes the specified **character device** to determine the file system type. It then contacts the local **pfs_mountd.rpc** program to register the specified *directory* as a valid mounted file system. **pfs_mountd.rpc** will reply with the address of the **pfsd.rpc** who will be handling all requests for files on that *directory*.

Remote mounts are very similar, except that both the local and remote mount daemons will be contacted. The remote mount daemon will supply the PFS server address, and the local mount daemon will be contacted to register the mount.

pfs_umount unmounts a currently mounted PFS file system, which can be specified as either a *directory* or a *filesystem*.

pfs_umount contacts the local mount daemon to determine what actions should be taken to perform the unmount. If the file system was originally remotely mounted, the remote mount daemon is informed of the unmount, and the file system is unmounted. Otherwise, it is simply unmounted.

pfs_mount and **pfs_umount** maintain a table of mounted file systems in **/etc/pfs_mtab**, described in *pfs_fstab*(5). If invoked without an argument, **pfs_mount** displays the contents of this table. If invoked with either a *filesystem* or a *directory* only, **pfs_mount** searches the file **/etc/pfs_fstab** for a matching entry, and mounts the file system indicated in that entry on the indicated directory.

If a user unmounts a PFS file system with the **umount** program, or interrupts the **pfs_umount** program before it has completed processing, the PFS daemons may leave the mount device open after the file system is no longer accessible. To clear these problems, use the **-c** flag for **pfs_umount**.

PFS expects a **character device** to be used for mounts, not a block device. Use of a block device with PFS is not supported.

pfs_mount Options

-v	Verbose. Display a message indicating each file system being mounted.
-a	All. Mount all file systems described in the **/etc/pfs_fstab** file.
-f	Fake an **/etc/mnttab** entry, but do not actually mount any file systems. Note: This option has no effect on HP-UX 10.30 or later.
-n	Mount the file system without making an entry in **/etc/mnttab**. Note: This option has no effect on HP-UX 10.30 or later.
-x *xlat*	Filename translation options. Any combination can be specified, although some combinations do not make sense (i.e. **dot_version** and **no_version**).

no_version	will suppress the printing of the version number (and semicolon) at the end of ISO 9660 and High Sierra filenames.
dot_version	replaces the version number (and semicolon) with a period followed by the version number.
lower_case	Converts upper to lower case on all file (and directory) names.
unix	Shorthand for **no_version** and **lower_case**.

-t *type* Force the CD-ROM to be mounted as the specified type, if possible. Accepted types are:

iso9660	will cause the mount program to attempt to mount the CD-ROM image using the ISO 9660 specifications. If the CD image is not ISO 9660 compatible, the mount fails. Note that if the CD image is also Rock Ridge compliant, and the **-t iso9660** option is not specified, the CD-ROM image will be mounted with Rock Ridge extensions enabled.
hsfs	will cause the mount program to attempt to mount the CD-ROM image using the High Sierra specifications. If the CD image is not **hsfs** compatible, the mount fails.
rrip	will cause the mount program to attempt to mount the CD-ROM image using the Rock Ridge Interchange specifications. If the CD image is not RRIP compatible, the mount fails. Note, that if the CD-ROM image supports the Rock Ridge Interchange Protocol, and the CD-ROM image is mounted with **rrip**, the translation options are suppressed.

Note that these get entered into the **/etc/pfs_mtab** and **/etc/pfs_fstab** with a **pfs-** preceding the type.

-o *options* Specify file system *options* as a list of comma-separated words from the list below.

options valid on *all* file systems:

ro	Even if not specified, the read-only option is implied.
suid | **nosuid**	SetUID execution allowed or disallowed.
bg | **fg**	If the first attempt fails, retry in the background, or, in the foreground.
retry=*n*	The number of times to retry the mount operation.
rsize=*n*	Set the read buffer size to *n* bytes.
timeo=*n*	Set the PFS timeout to *n* tenths of a second.
retrans=*n*	The number of PFS retransmissions.
soft | **hard**	Return an error if the server does not respond, or continue the retry request until the server responds.
intr	Allow keyboard interrupts on hard mounts.

The defaults are:

```
suid,fg,retry=10000,timeo=7,rsize=2048,retrans=3,hard
```

options specific to **iso9660** and **hsfs** file systems:

xlat=xlat_flags

xlat_flags is a colon (:) separated list of translation options. Currently supported are **no_version**, **dot_version**, **lower_case**, and **unix**. They allow you to perform the same translations options the *-x* flag does. The *-x* flag remains for backward compatibility. It is suggested that you use the *xlat=* option flag as they can be placed in the **/etc/pfs_fstab** file.

pfs_umount Options

-v Verbose. Display a message indicating each file system as it is unmounted.

-a All. Unmount all PFS mounted file systems.

-c Close. Instruct the PFS daemons to close the given file system, but do not attempt to umount the file system. This is useful when the file system has already been unmounted, but the PFS daemons still have the source **character device** open.

pfs_mount CONFIGURATIONS

Background vs. Foreground

Filesystems mounted with the **bg** option indicate that **pfs_mount** is to retry in the background if the server's mount daemon (see *pfs_mountd*(1M)) does not respond. **pfs_mount** retries the request up to the count specified in the **retry=***n* option. Once the file system is mounted, each PFS request made in the kernel waits **timeo=***n* tenths of a second for a response. If no response arrives, the time-out is multiplied by **2** and the request is retransmitted. When the number of retransmissions has reached the number specified in the **retrans=***n* option, a file system mounted with the **soft** option returns an error on the request; one mounted with the **hard** option prints a warning message and continues to retry the request.

Interrupting Processes With Pending PFS Requests

The **intr** option allows keyboard interrupts to kill a process that is hung while waiting for a response on a hard-mounted file system.

Attributes Cache

The server's attribute cache retains file attribute information on requests that have been made. This provides faster access to entries which have previously been decoded.

Lookup Cache

The **Lookup Cache** holds information about the sequential nature of the directory entries. This cache stores the location of the next directory entry. When a request comes in for a directory entry, if the preceding directory entry had been accessed earlier, this location is examined first to see if the directory entry being requested matches the directory entry at that location.

Block Cache

This cache holds raw 8k blocks of recently accessed data.

EXAMPLES

To mount a CD-ROM disk:

```
pfs_mount /dev/rdsk/c0t6d0 /cd-rom
```

To mount a remote file system:

```
pfs_mount serv:/cd-rom /cd-rom
```

To fake an entry for iso9660 on /cd-rom:

```
pfs_mount -f -t iso9660 /dev/rdsk/c0t6d0 /cd-rom
```

To hard mount a remote file system:

```
pfs_mount -o hard serv:/cd-rom /cd-rom
```

AUTHOR

pfs_mount was developed by Young Minds, Inc.

FILES

/etc/mnttab	table of mounted file systems
/etc/pfs_fstab	table of PFS file systems
/etc/pfs_mtab	table of mounted PFS file systems

p

SEE ALSO

pfs_fstab(5), pfs_mountd(1M), pfsd(1M).

BUGS

If the directory on which a file system is to be mounted is a symbolic link, the file system is mounted on *the directory to which the symbolic link refers*, rather than being mounted on top of the symbolic link itself.

On Pioneer six disc changers (and perhaps other drives) if you mount the file system using the block device driver, the Pioneer returns information to the driver indicating there is no data, causing the mount to fail. Either mount the file system again (which will should succeed), or use the raw device driver.

NAME

pfs_mountd, pfs_mountd.rpc - PFS mount request server

SYNOPSIS

/usr/etc/pfs_mountd

DESCRIPTION

This program is available with the *Portable File System Package (PFS)*. **pfs_mountd** is an RPC server that answers file system mount requests. In the case of remote mount requests, it reads the file **/etc/pfs_xtab**, described in *pfs_exports*(5), to determine which file systems are available for mounting by which machines.

It is recommended that the **pfs_mountd** daemon be invoked by *rc*(1M). It must be invoked in the background.

The **pfs** mount daemon is composed of two programs: **pfs_mountd** and **pfs_mountd.rpc**. The **pfs_mountd.rpc** program should **not** be run directly. It is invoked by the **pfs_mountd** program.

The mount daemon assigns servers to mounted file systems in a round-robin fashion. For example, if there are four pfs daemons, and four **pfs_mount**'s are performed, each daemon will be serving a different mount.

Options

-v Verbose. Show version number, etc.

AUTHOR

pfs_mountd was developed by Young Minds, Inc.

FILES

/etc/pfs_xtab

SEE ALSO

pfs_exports(5), rc(1M).

NAME

pfsd, pfsd.rpc - PFS daemon

SYNOPSIS

pfsd [*nservers*] [**-v**] [**-o** *options*]

DESCRIPTION

pfsd starts the daemons that handle client filesystem requests. *nservers* is the number of file system server daemons to start. This number should be based on the load expected on this server. The load is defined by the number of mounted file systems.

Mounts are distributed in a round-robin fashion to the **pfsd** daemons.

It is recommended that the **pfsd** daemon be invoked by *rc*(1M). It must be invoked in the background.

The **PFS**daemon is composed of two programs: **pfsd** and **pfsd.rpc**. The **pfsd.rpc** program should **not** be run directly. It is invoked by the **pfsd** program.

Options

-v	Verbose. Show version number, etc.
-o *options*	Specify filesystem *options* using a comma-separated list from the following:

acsize=*n*	The number of entries to keep in the attribute cache (1390 bytes per entry).
bcsize=*n*	The number of entries to keep in the block cache (8244 bytes per entry).
lcsize=*n*	The number of entries to keep in the lookup cache (56 bytes per entry).

The defaults are: **acsize=200,bcsize=25,lcsize=100**

Attributes Cache

The server's attribute cache retains file attribute information on requests that have been made. This provides faster access to entries which have previously been decoded.

Lookup Cache

The lookup cache holds information about the sequential nature of the directory entries. This cache stores the location of the next directory entry. When a request comes in for a directory entry, if the preceding directory entry had been accessed earlier, this location is examined first to see if the directory entry being requested matches the directory entry at that location.

Block Cache

This cache holds raw 8k blocks of recently accessed data.

EXAMPLES

To start a pfs daemon with a 400 entry attribute cache:

```
pfsd -o acsize=400 &
```

To start 4 pfs daemons with the default cache sizes:

```
pfsd 4 &
```

WARNINGS

It is not a good idea to have the cache sizes of the **pfsd** exceed the amount of physical memory (or actually a small portion thereof). If the **pfsd** spends excessive amounts of time swapping to and from disk, the benefits of the caching are diminished.

Specifying cache which consume more virtual memory than available will cause the daemon to die with a virtual memory error.

AUTHOR

pfsd was developed by Young Minds, Inc.

SEE ALSO

pfs_mountd(1M).

NAME

ping - send ICMP Echo Request packets to network host

SYNOPSIS

ping [**-oprv**] [**-i** *address*] [**-t** *ttl*] *host* [**-n** *count*]

ping [**-oprv**] [**-i** *address*] [**-t** *ttl*] *host* *packet-size* [[**-n**] *count*]

DESCRIPTION

The **ping** command sends ICMP Echo Request (ECHO_REQUEST) packets to *host* once per second. Each packet that is echoed back via an ICMP Echo Response packet is written to the standard output, including round-trip time.

ICMP Echo Request datagrams ("pings") have an IP and ICMP header, followed by a **struct timeval** (see *gettimeofday*(2)) and an arbitrary number of "pad" bytes used to fill out the packet. The default datagram length is 64 bytes, but this can be changed by using the *packet-size* option.

Options

The following options and parameters are recognized by **ping**:

-i *address* If *host* is a multicast address, send multicast datagrams from the interface with the local IP address specified by *address* in “dot” notation (see *inet*(3N)). If the **-i** option is not specified, multicast datagrams are sent from the default interface, which is determined by the route configuration.

-o Insert an IP Record Route option in outgoing packets, summarizing routes taken when the command terminates.

It may not be possible to get the round-trip path if some hosts on the route taken do not implement the IP Record Route option. A maximum of 9 Internet addresses can be recorded due to the maximum length of the IP option area.

-p The new Path MTU information is displayed when a ICMP "Datagram Too Big" message is received from a gateway. The **-p** option must be used in conjunction with a large *packetsize* and with the **-v** option.

-r Bypass the normal routing tables and send directly to a host on an attached network. If the host is not on a directly-connected network, an error is returned. This option can be used to ping the local system through an interface that has no route through it, such as after the interface was dropped by **gated** (see *gated*(1M)).

-t *ttl* If *host* is a multicast address, set the time-to-live field in the multicast datagram to *ttl*. This controls the scope of the multicast datagrams by specifying the maximum number of external systems through which the datagram can be forwarded.

If *ttl* is zero, the datagram is restricted to the local system. If *ttl* is one, the datagram is restricted to systems that have an interface on the network directly connected to the interface specified by the **-i** option. If *ttl* is two, the datagram can forwarded through at most one multicast router; and so forth. *Range*: zero to 255. The default value is 1.

-v Verbose output. Show ICMP packets other than Echo Responses that are received.

host Destination to which the ICMP Echo Requests are sent. *host* can be a hostname or an Internet address. All symbolic names specified for *host* are looked up by using **gethostbyname()** (see *gethostent*(3N)). If *host* is an Internet address, it must be in "dot" notation (see *inet*(3N)).

If a system does not respond as expected, the route might be configured incorrectly on the local or remote system or on an intermediate gateway, or there might be some other network failure. Normally, *host* is the address assigned to a local or remote network interface.

If *host* is a broadcast address, all systems that receive the broadcast should respond. Normally, these are only systems that have a network interface on the same network as the local interface sending the ICMP Echo Request.

If *host* is a multicast address, only systems that have joined the multicast group should respond. These may be distant systems if the **-t** option is specified, and there is a multicast router on the network directly connected to the interface specified by the **-i** option.

packet-size The size of the transmitted packet, in bytes. By default (when *packet-size* is not specified), the size of transmitted packets is 64 bytes. The minimum value allowed for *packet-size* is 8 bytes, and the maximum is 4095 bytes. If *packet-size* is smaller than 16 bytes, there is not enough room for timing information. In that case, the round-trip times are not displayed.

count The number of packets **ping** will transmit before terminating. *Range*: zero to 2147483647. The default is zero, in which case **ping** sends packets until interrupted.

When using **ping** for fault isolation, first specify a local address for *host* to verify that the local network interface is working correctly. Then specify host and gateway addresses further and further away to determine the point of failure. **ping** sends one datagram per second, and it normally writes one line of output for every ICMP Echo Response that is received. No output is produced if there are no responses. If an optional *count* is given, only the specified number of requests is sent. Round-trip times and packet loss statistics are computed. When all responses have been received or the command times out (if the *count* option is specified), or if the command is terminated with a **SIGINT**, a brief summary is displayed.

This command is intended for use in testing, managing and measuring network performance. It should be used primarily to isolate network failures. Because of the load it could impose on the network, it is considered discourteous to use **ping** unnecessarily during normal operations or from automated scripts.

AUTHOR

ping was developed in the Public Domain.

FILES

/etc/hosts

SEE ALSO

gethostent(3N), inet(3N).

NAME

power_onoff - timed, automatic system power on, and power off

SYNOPSIS

/usr/sbin/power_onoff -n

/usr/sbin/power_onoff *time* [*date*] [[**next** | **+***increment*] *time_designation*]

DESCRIPTION

power_onoff instructs the UPS monitor (**ups_mond**) to shut down the system, and optionally informs the monitor when to power on the system again. The UPS monitor in turn instructs the uninterruptible power source (UPS) when to turn the power off and on. The UPS monitor then proceeds to shut down the system. The time to restart the system (power on) is specified with **power_onoff** command-line arguments.

Some UPS units limit the time that can elapse between the time the power is turned off and the time it is turned back on. Please see your UPS documentation for information about limitations.

power_onoff requires a UPS that is supported by the UPS monitor (see *ups_mond*(1M)).

Command Line Arguments

The **power_onoff** command has two forms, and recognizes the following arguments:

-n No power on. Causes the system to be shutdown and not be powered back on.

time Can be specified as one, two, or four digits. One- and two-digit numbers represent hours; four digits represent hours and minutes. *time* can also be specified as two numbers separated by a colon (**:**), single quote (**'**), the letter "h" (**h**), a period (**.**), or comma (**,**). A suffix **am** or **pm** can be appended. Otherwise a 24-hour clock time is understood. For example, **0815**, **8:15**, **8'15**, **8h15**, **8.15**, and **8,15** are read as 15 minutes after 8 in the morning. The suffixes **zulu** and **utc** can be used to indicate Coordinated Universal Time. The special names **noon**, **midnight**, **now**, and **next** are also recognized.

date Can be specified as either a day of the week (fully spelled out or abbreviated) or a date consisting of a day, a month, and optionally a year. The day and year fields must be numeric, and the month can be fully spelled out, abbreviated, or numeric. These three fields can be in any order, and be separated by punctuation marks such as slash (**/**), hyphen (**-**), period (**.**), or comma (**,**). The years 00-68 would be interpreted as 2000-2068 and 69-99 would be 1969-1999. Two special "days", **today** and **tomorrow**, are also recognized. If no *date* is given, **today** is assumed if the given time is greater than the current time; **tomorrow** is assumed if it is less. If the given month is less than the current month (and no year is given), next year is assumed.

next or **+***increment* If followed by a *time_designation* of **minutes**, **hours**, **days**, **weeks**, **months**, or **years**, lets the user startup the system when the specified *time_designation* has elapsed. A numerical operator, **+***increment*, enables the user to schedule the startup several hours, days, weeks, months, or years in advance (see EXAMPLES). Using the argument **next** is equivalent to using an *increment* of **+1**. Both plural and singular forms of *time_designation* are accepted.

EXTERNAL INFLUENCES

International Code Set Support

Single- and multi-byte character code sets are supported.

RETURN VALUE

Exit code 0 is returned upon successful completion, otherwise non 0 is returned.

DIAGNOSTICS

power_onoff issues diagnostic messages when it encounters syntax errors and out-of-range times.

EXAMPLES

To startup the system at 5:00 am next Tuesday, use

```
power_onoff 5am Tuesday next week
```

To startup the system at 5:30 am tomorrow, use

```
power_onoff 5:30 tomorrow
```

To make your system startup each weekday at 7:30 am and shutdown at 5:30 pm each week day, use **crontab** to execute the first entry on Monday through Thursday and the second entry on Friday (see *crontab*(1)).

```
power_onoff 7:30 tomorrow
power_onoff 7:30 Monday
```

To startup the system at 8:15 on January 24, use

```
power_onoff 0815 Jan 24
```

To startup the system at 5:15 on January 24, use

```
power_onoff 5:15 Jan 24
```

To startup the system at 9:30 tomorrow, use

```
power_onoff 9:30am tomorrow
```

To startup the system 24 hours from now, use

```
power_onoff now + 1 day
```

To shutdown the system and not start it up, use

```
power_onoff -n
```

WARNINGS

Jobs can be submitted up to 2037. If jobs were submitted any later than 2037, an error message will display "BAD DATE".

Some UPS units limit the time that can elapse between the time the power is turned off and the time it is turned back on. Please see your UPS documentation for information about limitations.

If the *date* argument begins with a number and the *time* argument is also numeric (and without suffix), the *time* argument should be a four-digit number that can be correctly interpreted as hours and minutes.

Do not use both **next** and **+** *increment* within a single **power_onoff** command; only the first operator is accepted and the trailing operator is ignored. No warning or error is produced.

The power cord must be disconnected before servicing the unit.

AUTHOR

power_onoff was developed by HP.

FILES

/var/tmp/timed_off	fifo for communicating with ups_mond.

SEE ALSO

at(1), cron(1M), crontab(1), queuedefs(4), proto(4), kill(1), sam(1M), ups_mond(1M).

NAME

pscan - scan an HP SCSI disk array LUN for parity consistency

SYNOPSIS

pscan -s *system_state device_file*

DESCRIPTION

pscan is a front end script to **scn** designed to be called during system bootup and shutdown. It stores information on the disk array used to indicate improper system shutdown. If the system was not properly shutdown, a parity scan is initiated when the system boots. The values of 0 (operating system booting), and 1 (operating system shutdown) are expected for *system_state*. *device_file* refers to the device file associated with the selected disk array. If multiple hosts (initiators) are connected to the disk array, the file /etc/hpC2400/pscan.initiators needs to be created. This file will contain an integer value from 1 to 8. If the file is not created, pscan will assume that only one host (initiator) is connected to the disk array.

RETURN VALUE

pscan returns the following values:

0	Successful completion.
-1	Command failed (an error occurred).

DIAGNOSTICS AND ERRORS

Errors can originate from problems with:

- **pscan**
- SCSI (device level) communications
- system calls

Error messages generated by pscan:

usage: pscan -s <system_state> <special>
An error in command syntax has occurred. Enter command again with all required arguments, in the order shown.

pscan: LUN # too big
The LUN number, which is derived from the device file name, is out of range.

pscan: Not a raw file
Utilities must be able to open the device file for raw access.

pscan: LUN does not exist
The addressed LUN is not configured, and thus is not known to the array controller.

pscan: Not an HP SCSI disk array
The device being addressed is not an HP SCSI disk array.

SCSI (device level) communication errors:

Sense data associated with the failed operation is printed.

Error messages generated by system calls:

pscan uses the following system calls:

malloc(), **free()**, **stat()**, **open()**, **close()**, **fopen()**, **fclose()**, **read()**, **write()**, and **ioctl()**.

Documentation for these HP-UX system calls contains information about the specific error conditions associated with each call. **pscan** does not alter the value of **errno**. The interpretation of **errno** for printing purposes is performed by the system utility **strerror()**.

EXAMPLES

To call pscan on the LUN **/dev/rdsk/c2t0d2** prior to a system shutdown on a series 800:

```
pscan -s 1 /dev/rdsk/c2t0d2
```

To call pscan on the LUN **/dev/rdsk/c2t0d2** during a system bootup on a series 700:

```
pscan -s 0 /dev/rdsk/c2t0d2
```

p

DEPENDENCIES

The HP C2425 and HP C2427 disk arrays are only supported on Series 700 systems running HP-UX version 9.0X.

The HP C2430 disk array is supported on Series 700 and 800 systems running HP-UX versions 9.0X and 10.0X.

AUTHOR

pscan was developed by HP.

NAME

pvchange - change characteristics and access path of physical volume in LVM volume group

SYNOPSIS

/usr/sbin/pvchange [**-A** *autobackup*] **-s** *pv_path*

/usr/sbin/pvchange [**-A** *autobackup*] **-S** *autoswitch pv_path*

/usr/sbin/pvchange [**-A** *autobackup*] **-x** *extensibility pv_path*

/usr/sbin/pvchange [**-A** *autobackup*] **-t** *IO_timeout pv_path*

/usr/sbin/pvchange [**-A** *autobackup*] **-z** *sparepv pv_path*

Remarks

pvchange cannot be performed if the volume group is activated in shared mode.

DESCRIPTION

The **pvchange** command changes the characteristics and access path of a physical volume (*pv_path*) in a volume group.

For multiported devices accessed via multiple paths, **pvchange** may be used to customize the circumstances that may cause LVM to automatically switch from one path to another, or when LVM will switch back to a prior path which failed when it is available again (generally described as the physical volume's *autoswitch* behavior). **pvchange** also permits you to switch manually to a specific path to the physical volume.

pvchange sets the allocation permission to add physical extents to the physical volume.

If you have installed the optional HP MirrorDisk/UX software, you can use the **-z** option to designate a spare physical volume to be used to replace an existing physical volume within a volume group when mirroring is in effect, in the event the existing physical volume fails.

Options and Arguments

pvchange recognizes the following options and arguments.

pv_path The block device path name of a physical volume.

-A *autobackup* Set automatic backup for this invocation of this command. *autobackup* can have one of the following values:

- **y** Automatically back up configuration changes made to the logical volume. This is the default.

  After this command executes, the **vgcfgbackup** command (see *vgcfgbackup*(1M)) is executed for the volume group to which the logical volume belongs.

- **n** Do not back up configuration changes this time.

-s Immediately begin accessing the associated physical volume named by *pv_path*.

-S *autoswitch* This option specifies the *autoswitch* behavior for multiported physical volumes accessed via multiple paths. It has no effect for physical volumes without alternate paths. *autoswitch* option may be set to one of the following values:

- **y** LVM is directed to automatically switch from the path it is using whenever a better path to the physical volume is available. LVM will switch paths when a better path recovers (after it had failed earlier), or if the current path fails and another path is available. This is the default.

- **n** LVM is directed to automatically switch to using the best available path only when the path currently in use is unavailable. LVM will continue using a specific path for the physical volume as long as it works, regardless of whether another better path recovers from a failure.

-x *extensibility* Set the allocation permission to add physical extents to the physical volume *pv_path*. *extensibility* can have the following values:

p

- **y** Allow allocation of additional physical extents on the physical volume. This is the default.
- **n** Prohibit allocation of additional physical extents on the physical volume. However, logical volumes residing on the physical volume are accessible.

-t *IO_timeout* Set *IO_timeout* for the physical volume to the number of seconds indicated. An *IO_timeout* value of zero (0) causes the system to use the default value supplied by the device driver associated with the physical device. *IO_timeout* is used by the device driver to determine how long to wait for disk transactions to complete before concluding that an IO request can not be completed (and the device is offline or unavailable).

-z *sparepv* This option requires the installation of the optional HP MirrorDisk/UX software. It allows you to change the physical volume specified by *pv_path* into a spare physical volume for its volume group, or change the specified spare physical volume back into a regular physical volume for this volume group. No physical extents from a spare physical volume will be available as part of the "free" pool of extents in the volume group. A spare physical volume will only be used in the event that another physical volume within this volume group becomes unavailable (fails). *sparepv* can have one of the following values:

- **y** Change the specified physical volume to be a "stand-by" spare for its volume group. The specified physical volume must not have extents allocated on it (i.e., no logical volumes residing on it) at the time this command is issued. A stand-by spare physical volume will only be used in the event of a failure of another physical volume -- prior to such a failure, no logical volume is allowed to reside on it.
- **n** Change the specified spare physical volume back into a regular physical volume. If the physical volume was a stand-by spare, then all of the disk space associated with it will be immediately available for use by logical volumes. If the physical volume is an "active" spare, that is, it was previously a stand-by spare but then took over for a failed physical volume, it will simply mark the physical volume as a regular member of its volume group and the logical volumes residing on it will remain unchanged.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages are displayed.

If **LANG** is not specified or is null, it defaults to **C** (see *lang*(5)).

If any internationalization variable contains an invalid setting, all internationalization variables default to **C** (see *environ*(5)).

EXAMPLES

Prohibit the allocation of additional physical extents to a physical volume:

```
pvchange -x n /dev/dsk/c0t0d0
```

Allow the allocation of additional physical extents to a physical volume:

```
pvchange -x y /dev/dsk/c0t0d0
```

Only switch paths when the current path is unavailable. Do not switch back to a prior path which had failed and has recovered, when the current path works:

```
pvchange -S n /dev/dsk/c0t0d0
```

Switch paths whenever a better path becomes available again after a failure, even if the current path is fine:

```
pvchange -S y /dev/dsk/c0t0d0
```

Manually switch a physical volume to use another controller path:

```
pvchange -s /dev/dsk/c2t0d2
```

Set the *IO_timeout* value for a physical volume to 60 seconds:

```
pvchange -t 60 /dev/dsk/c2t0d2
```

Set the *IO_timeout* value for a physical volume to zero (0) to use the driver default:

```
pvchange -t 0 /dev/dsk/c2t0d2
```

Change the (empty) physical volume to become a stand-by spare for the volume group:

```
pvchange -z y /dev/dsk/c2t0d2
```

Change the (active or stand-by) spare physical volume to become a regular member of the volume group:

```
pvchange -z n /dev/dsk/c2t0d2
```

SEE ALSO

pvdisplay(1M).

NAME

pvck - check or repair a physical volume in LVM volume group

SYNOPSIS

/usr/sbin/pvck -y *pv_path*

/usr/sbin/pvck -n *pv_path*

DESCRIPTION

Note: Currently **pvck** is only capable of detecting bad checksums caused by a forward system migration after a backward system migration. It should not be used in other situations.

The **pvck** command examines and repairs LVM data structures on a raw disk (*pv_path*) in a volume group.

Options and Arguments

pvck recognizes the following options and arguments.

-y	Repair problems found.
-n	Report, but do not repair problems.
pv_path	The raw device path name of a physical volume.

RETURN VALUE

pvck returns the following values

0	Either no problems were found or all problems were corrected.
1	Unable to repair.

EXAMPLES

Examine LVM checksums on **/dev/rdsk/c0t6d0** without modifying anything:

```
pvck -n /dev/rdsk/c0t6d0
```

Repair LVM checksums on **/dev/rdsk/c0t6d0** if necessary:

```
pvck -y /dev/rdsk/c0t6d0
```

WARNINGS

pvck should only be run on a device whose volume group has not been activated.

It is designed to repair the root device or devices while the system is booted in maintenance mode ("**hpux -lm**", see *hpux*(1M)).

AUTHOR

pvck was developed by HP.

NAME

pvcreate - create physical volume for use in LVM volume group

SYNOPSIS

/usr/sbin/pvcreate [**-b**] [**-B**] [**-d** *soft_defects*] [**-s** *disk_size*] [**-f**] [**-t** *disk_type*] *pv_path*

DESCRIPTION

The **pvcreate** command initializes a direct access storage device (a raw disk device) for use as a physical volume in a volume group.

If *pv_path* contains a file system and the **-f** option is not specified, **pvcreate** asks for confirmation. The request for confirmation avoids accidentally deleting a file system.

The operation is denied if *pv_path* belongs to another volume group. Only physical volumes not belonging to other volume groups can be created. The operation is also denied if *pv_path* refers to a disk device already under the control of the VERITAS Volume Manager.

After using **pvcreate** to create a physical volume, use **vgcreate** to add it to a new volume group or **vgextend** to add it to an existing volume group (see *vgcreate*(1M) and *vgextend*(1M)).

Disks cannot be added to a volume group until they are properly initialized by **pvcreate**.

pv_path can be made into a bootable disk by specifying the **-B** option, which reserves space on the physical volume for boot-related data. This is a prerequisite for creating root volumes on logical volumes. Refer to *mkboot*(1M) and *lif*(4) for more information.

Options and Arguments

pvcreate recognizes the following options and arguments:

pv_path	The character (raw) device path name of a physical volume.
-b	Read from standard input the numbers that correspond to the indexes of all known bad blocks on the physical volume, *pv_path*, that is being created. Specify the indexes using decimal, octal, or hexadecimal numbers in standard C-language notation, with numbers separated by newline, tab, or formfeed characters. If this option is not used, **pvcreate** assumes that the physical volume contains no bad blocks.
-B	Make a bootable physical volume (i.e., a system disk).
-d *soft_defects*	Specify the minimum number of bad blocks that LVM should reserve in order to perform software bad block relocation. This number can be no larger than 7039. If not specified, one block is reserved for each 8K of data blocks.
-s *disk_size*	Effective size of the physical volume to be created, specified in number of physical sectors.
-f	Force the creation of a physical volume (thus deleting any file system present) without first requesting confirmation.
-t *disk_type*	Retrieve configuration information about the physical volume from the file **/etc/disktab**. *disk_type* specifies the device (for example, hp7959S). The *disk_type* only needs to be specified when **pvcreate** fails to get the size from the underlying disk driver. If the driver successfully returns the size of the device, *disk_type* is ignored.

p

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages are displayed.

If **LANG** is not specified or is null, it defaults to "C" (see *lang*(5)).

If any internationalization variable contains an invalid setting, all internationalization variables default to "C" (see *environ*(5)).

EXAMPLES

Create a physical volume on raw device **/dev/rdsk/c1t0d0**, and force the creation without confirmation:

```
pvcreate -f /dev/rdsk/c1t0d0
```

Create a physical volume on raw device **/dev/rdsk/c1t0d0**, specifying that a bad blocks list (7, 13, 95, and 133) must be read from standard input:

```
echo 7 13 95 133 | pvcreate -b /dev/rdsk/c1t0d0
```

FILES

/etc/disktab	Disk geometry and disk partition characteristics for all disk devices on the system

WARNINGS

Check the manufacturer's listing or run diagnostics testing for bad blocks on the device prior to creating a physical volume. If bad blocks are present, use the **-b** option when creating the physical volume.

SEE ALSO

mkboot(1M), vgcreate(1M), vgextend(1M), lif(4).

NAME

pvdisplay - display information about physical volumes within LVM volume group

SYNOPSIS

/usr/sbin/pvdisplay [**-v**] [**-b** *BlockList*] *pv_path* ...

DESCRIPTION

The **pvdisplay** command displays information about each physical volume specified by a *pv_path* parameter.

Options

pvdisplay recognizes the following options:

pv_path The block device path name of a physical volume.

-v For each physical volume, display the logical volumes that have extents allocated on the physical volume and the usage of all the physical extents.

-b *BlockList*
For each block in *BlockList*, display information about the block. *BlockList* is a comma separated list of blocks in **DEV_BSIZE** units.

Display Without –v Option

If you omit the **-v** option, **pvdisplay** displays the characteristics of each physical volume specified by *pv_path*:

--- Physical volumes ---

PV Name The block device path name of the physical volume

VG Name The path name of the volume group

PV Status State of the physical volume (*NOTE*: spare physical volumes are only relevant if you have installed HP MirrorDisk/UX software):

available The physical volume is available and is not a spare physical volume.

available/data spared
The physical volume is available. However, its data still resides on an active spare.

available/active spare
The physical volume is available and is an active spare physical volume. (An active spare is a spare that has taken over for a failed physical volume.)

available/standby spare
The physical volume is a spare, "standing by" in case of a failure on any other physical volume in this volume group. It can only be used to capture data from a failed physical volume.

unavailable The physical volume is unavailable and is not a spare physical volume.

unavailable/data spared
The physical volume is unavailable. However, its data now resides on an active spare, and its data is available if the active spare is available.

unavailable/active spare
The physical volume is unavailable and it’s an active spare. Thus, the data on this physical volume in unavailable.

unavailable/standby spare
The physical volume is a spare, "standing by" that is not currently available to capture data from a failed physical volume.

`Allocatable`	Allocation permission for the physical volume
`VGDA`	Number of volume group descriptors on the physical volume
`Cur LV`	Number of logical volumes using the physical volume
`PE Size (Mbytes)`	Size of physical extents on the volume, in megabytes (MB)
`Total PE`	Total number of physical extents on the physical volume
`Free PE`	Number of free physical extents on the physical volume
`Allocated PE`	Number of physical extents on the physical volume that are allocated to logical volumes
`Stale PE`	Number of physical extents on the physical volume that are not current
`IO Timeout`	The IO timeout used by the disk driver when accessing the physical volume. A value of **`default`**, indicates that the driver default IO timeout is being used.
`Spared from PV`	If the physical volume represents an active spare, this field will show the name of the failed physical volume whose data now resides on this spare. This information can be used to manually move the data back to the original physical volume, once it has been repaired. (See *pvmove*(1M)*).* If it cannot be determined which physical volume that the data came from, this field will instead display **`Missing PV`**. A missing PV would indicate that when the volume group was last activated or reactivated (see *vgchange*(1M)*),* the "failed" physical volume was not able to attach to the volume group.
`Spared to PV`	If the physical volume represents a failed physical volume, this field will show the name of the active spare physical volume that now contains the data that originally resided on this volume. This information can be used to manually move the data back to the original physical volume (see *pvmove*(1M)*)* once it has been repaired.

Display With −v Option

If **`-v`** is specified, **`pvdisplay`** lists additional information for each logical volume and for each physical extent on the physical volume:

`--- Distribution of physical volume ---`

The logical volumes that have extents allocated on *pv_path*, displayed in column format:

`LV Name`	The block device path name of the logical volume which has extents allocated on *pv_path*.
`LE of LV`	Number of logical extents within the logical volume that are contained on this physical volume
`PE for LV`	Number of physical extents within the logical volume that are contained on this physical volume

`--- Physical extents ---`

The following information for each physical extent, displayed in column format:

`PE`	Physical extent number
`Status`	Current state of the physical extent: **`free`**, **`used`**, or **`stale`**
`LV`	The block device path name of the logical volume to which the extent is allocated
`LE`	Index of the logical extent to which the physical extent is allocated

Display With −b Option

If **`-b`** is specified, **`pvdisplay`** lists additional information for each block specified in *BlockList*.

`--- Block Mapping ---`

The use of blocks on *pv_path*, displayed in column format:

`Block`	The block number relative to the physical volume.
`Status`	The current status of the block: **`free`**, **`used`**, **`structure`**, **`spared`**, or **`unknown`**
`Offset`	The offset of the block relative to the logical volume.
`LV Name`	The block device path name of the logical volume to which the block is allocated.

EXTERNAL INFLUENCES

Environment Variables

`LANG` determines the language in which messages are displayed.

If **`LANG`** is not specified or is null, it defaults to "C" (see *lang*(5)).

If any internationalization variable contains an invalid setting, all internationalization variables default to "C" (see *environ*(5)).

EXAMPLES

Display the status and characteristics of a physical volume:

```
pvdisplay /dev/dsk/c1t0d0
```

Display the status, characteristics, and allocation map of a physical volume:

```
pvdisplay -v /dev/dsk/c2t0d0
```

SEE ALSO

lvdisplay(1M), pvchange(1M), vgdisplay(1M).

NAME

pvmove - move allocated physical extents from one LVM physical volume to other physical volumes

SYNOPSIS

/usr/sbin/pvmove [**-A** *autobackup*] [**-n** *lv_path*] *source_pv_path*
[*dest_pv_path* ... | *dest_pvg_name* ...]

Remarks

pvmove cannot be performed if the volume group is activated in shared mode.

DESCRIPTION

The **pvmove** command moves allocated physical extents and the data they contain from a source physical volume, *source_pv_path*, to one or more other physical volumes in the same volume group.

If a destination physical volume or physical volume group is not specified, all physical volumes in the volume group are available as destination volumes for the transfer. **pvmove** selects the proper physical volumes to be used in order to preserve the allocation policies of the logical volume involved.

To limit the transfer to specific physical volumes, specify the name of each physical volume directly with a *dest_pv_path* argument. Optionally, if physical volume groups are defined for the volume group, specify the physical volumes indirectly with one or more *dest_pvg_name* arguments.

source_pv_path must not appear as a *dest_pv_path*.

If *source_pv_path* is a member of a *dest_pv_path*, it is automatically excluded from being a destination physical volume.

pvmove succeeds only if there is enough space on the destination physical volumes to hold all the allocated extents of the source physical volume.

If you have installed HP MirrorDisk/UX on your system and *source_pv_path* is an "active spare" physical volume within a mirrored logical volume, once all of the data has been moved to *dest_pv_path*, the *source_pv_path* physical volume will be returned to a "stand-by" spare physical volume. This is how to "unspare" data once the original failed physical volume has been repaired and is available to receive data.

Options

pvmove recognizes the following options:

dest_pv_path	The block device path name of a physical volume. It cannot be the source physical volume. It must be in the same volume group as *source_pv_path*.
dest_pvg_name	The name of a physical volume group. It must be in the same volume group as *source_pv_path*.
source_pv_path	The block device path name of a physical volume.
-A *autobackup*	Set automatic backup for this invocation of this command. *autobackup* can have one of the following values: **y** Automatically back up configuration changes made to the physical volume. This is the default. After this command executes, the **vgcfgbackup** command (see *vgcfgbackup*(1M)) is executed for the volume group to which the physical volume belongs. **n** Do not back up configuration changes this time.
-n *lv_path*	Move only the physical extents allocated to the logical volume specified by *lv_path* that are located on the source physical volume specified by *source_pv_path*.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages are displayed.

If **LANG** is not specified or is null, it defaults to "C" (see *lang*(5)).

If any internationalization variable contains an invalid setting, all internationalization variables default to "C" (see *environ*(5)).

EXAMPLES

Move physical extents from **/dev/dsk/c1t0d0** to **/dev/dsk/c2t0d0** and **/dev/dsk/c3t0d0**:

```
pvmove /dev/dsk/c1t0d0 /dev/dsk/c2t0d0 /dev/dsk/c3t0d0
```

If physical volumes **/dev/dsk/c2t0d0** and **/dev/dsk/c3t0d0** are the only ones that belong to physical volume group **PVG0**, the same result can be achieved with the following command:

```
pvmove /dev/dsk/c1t0d0 PVG0
```

Move only the physical extents for logical volume **/dev/vg01/lvol2** that are currently on **/dev/dsk/c1t0d0** to **/dev/dsk/c2t0d0**:

```
pvmove -n /dev/vg01/lvol2 /dev/dsk/c1t0d0 /dev/dsk/c2t0d0
```

SEE ALSO

pvdisplay(1M), vgcfgbackup(1M).

NAME

pvremove - remove LVM data structure from a physical volume

SYNOPSIS

/sbin/pvremove *pv_path*

DESCRIPTION

The **pvremove** command clears the LVM data structure on a disk, so that it is no longer an LVM physical volume. The device may then be used by the file system or by other Volume Manager.

The operation is denied if *pv_path* is assigned to a volume group. The **pvremove** command only clears the LVM data structure on a disk if the disk does not belong to a volume group. This avoids accidentally removing a valid physical volume under a volume group. If the physical volume to be removed belongs to a volume group, use the **vgremove** command to first remove the volume group associated with the physical volume.

Arguments

pvremove recognizes the following arguments:

pv_path	The character (raw) device path name of a physical volume.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages are displayed.

If **LANG** is not specified or is null, it defaults to "C" (see *lang*(5)).

If any internationalization variable contains an invalid setting, all internationalization variables default to "C" (see *environ*(5)).

EXAMPLES

Remove LVM data structure on the physical volume **/dev/rdsk/c1t0d0**:

```
pvremove /dev/rdsk/c1t0d0
```

SEE ALSO

lvremove(1M), vgremove(1M).

NAME

pwck, grpck - password/group file checkers

SYNOPSIS

/usr/sbin/pwck [**-s**] [**-l**] [*file*]

/usr/sbin/grpck [*file*]

DESCRIPTION

pwck scans the default password file or *file* and reports any inconsistencies to standard error. The checks include validation of the number of fields, login name, user ID, group ID, and whether the login directory and optional program exist. In addition, if the root entry shows a program, it can only be one of: **/sbin/sh**, **/usr/bin/csh**, **/usr/bin/ksh**, or **/usr/bin/sh**. The criteria for validation is in *passwd*(4) and is described in the *Managing Systems and Workgroups* manual. The default password file is **/etc/passwd**.

grpck verifies all entries in the group file and reports any inconsistencies to standard error. This verification includes a check of the number of fields, group name, group ID, and whether all login names appear in the password file. The default group file is **/etc/group**.

Options

pwck recognizes the following options:

-s Check inconsistencies with the Protected Password database. It calls **authck -p**.

-l Check encrypted password lengths that are greater than 8 characters. If **NIS+** is running with Trusted mode, password lengths must not be longer than 8 characters.

DIAGNOSTICS

Group entries in **/etc/group** with no login names are flagged.

WARNINGS

Successful password file validation is not sufficient for proper system operation. To help maintain consistency with other system databases, editing of the password file with **vipw** is generally discouraged. Please use **sam**, **useradd**, **usermod**, **userdel**, **chfn**, **chsh** or **passwd** to edit **/etc/passwd**.

DEPENDENCIES

NFS:

pwck and **grpck** check only the local password and group files. The Network Information Service database for password and group files is not checked.

AUTHOR

pwck was developed by AT&T and HP.

FILES

/etc/group
/etc/passwd

SEE ALSO

authck(1M), vipw(1M), group(4), passwd(4).

STANDARDS CONFORMANCE

pwck: SVID2, SVID3

grpck: SVID2, SVID3

NAME

pwconv - update secure password facility

SYNOPSIS

pwconv

DESCRIPTION

If the secured password facility is already installed, **pwconv** updates the facility to reflect any changes made in the **/etc/passwd** file.

FILES

/etc/passwd
/tcb/files/auth/*/*

SEE ALSO

vipw(1M), pwck(1M).

NAME

pwgr_stat - Password and Group Hashing and Caching Statistics

SYNOPSIS

/usr/sbin/pwgr_stat

DESCRIPTION

pwgr_stat displays the current status of the **pwgrd** daemon process running on the system. It includes whether or not the daemon is running, how much activity is occurring, as well as statistics for each kind of request serviced by **pwgrd**. Request specific statistics include the number of request and the percent of requests handled by the cache and the hashtables used to service that request. A request may not have both a cache and a hashtable. Such requests will have a **-** for the corresponding hit rate. Requests where no answer was found are not counted in the hit rate.

The display is updated every 2 seconds. Use the **q** key to exit **pwgr_stat**. **pwgr_stat** verifies that **pwgrd** is accessible by issuing a **NULL** request to **pwgrd**, therefore the NULL request count will be increased as long as **pwgr_stat** is running.

FILES

/var/spool/pwgr/daemon	Daemon Unix domain socket.
/var/spool/pwgr/status	Daemon process status file.

AUTHOR

pwgr_stat was developed by the Hewlett-Packard Company.

SEE ALSO

pwgrd(1M).

p

NAME

pwgrd - Password and Group Hashing and Caching daemon

SYNOPSIS

/usr/sbin/pwgrd [**-d**] [**-l** *logfile*]

DESCRIPTION

pwgrd provides accelerated lookup of password and group information for libc routines like **getpwuid** and **getgrname**. **pwgrd** implements per request type caches and hashtables as appropriate. When the corresponding routine in libc is called, a request is issued to **pwgrd** via a Unix domain socket connection. **pwgrd** determines whether it can satisfy the request, returning the appropriate results to the requesting process.

Options

pwgrd recognizes the following options and command-line arguments:

-d	Debug mode. Do not become a daemon. Issue additional diagnostic messages. Instead of logging message via **syslog**, issue messages to stderr.
-l	*logfile* Logfile. In addition to logging via **syslog**, **pwgrd** will write log messages to *logfile*.

pwgrd modifies its behavior depending on whether or not the local machine is using some form of *NIS* for password or group information. When *NIS* or *NIS+* is being used, the hashtables corresponding to that service are not generated or consulted. Therefore only caching is provided for those requests.

FILES

/etc/rc.config.d/pwgr	Start up configuration variable. Set **PWGR** to **1** if you want **pwgrd** to start on reboot.
/var/spool/pwgr/*	Hash files, status file and daemon Unix domain socket.
/var/spool/sockets/pwgr/*	Client Unix domain sockets.

AUTHOR

pwgrd was developed by the Hewlett-Packard Company.

p

SEE ALSO

pwgr_stat(1M).

NAME

quot - summarize file system ownership

SYNOPSIS

/usr/sbin/quot [**-F** *FStype*] [**-V**] [**-cfhnv**] [**-o** *FSspecific-options*] *filesystem* ...

/usr/sbin/quot [**-F** *FStype*] [**-V**] [**-cfhnv**] **-a**

DESCRIPTION

The **quot** command displays the number of 1024-byte blocks in the named *filesystem* that are currently owned by each user. *filesystem* is either the name of the directory on which the file system is mounted or the name of the device containing the file system.

Options

quot recognizes the following options:

-F *FStype* Specify the file system type on which to operate (see *fstyp*(1M) and *fs_wrapper*(5)). If this option is not included on the command line, the file system type is determined from the file **/etc/fstab** by matching *filesystem* with an entry in that file. If there is no entry in **/etc/fstab**, then the file system type is determined from the file **/etc/default/fs**.

-V Echo the completed command line, but perform no other action. The command line is generated by incorporating the user-specified options and other information derived from **/etc/fstab**. This option allows the user to verify the command line.

-o *FSspecific-options*
Specify any options specific to the file system.

-a Generate a report for all mounted file systems.

-c Report size rather than user statistics. Generates histogram statistics in 3-column format:

Column 1:	File size in blocks. Sizes are listed in ascending order up to 499 blocks per file. Files occupying 499 or more blocks are counted together on a single line as 499-block files (but column 3 is based on actual number of blocks occupied).
Column 2:	Number of files of size indicated in column 1.
Column 3:	Cumulative total blocks occupied by files counted in current plus all preceding lines.

Use of this option overrides the **-f** and **-v** options.

-f Display number of files and space occupied by each user.

-h Calculate the number of blocks in the file based on file size rather than actual blocks allocated. This option does not account for sparse files (files with holes in them).

-n Accept data from the **ncheck** command (see *ncheck*(1M)) as input. Run the pipeline:

ncheck *device* **| sort +0n | quot -n** *filesystem*

to produce a list of all files and their owners.

-v Display three columns containing the number of blocks not accessed in the last 30, 60, and 90 days.

AUTHOR

quot was developed by the University of California, Berkeley, Sun Microsystems, Inc., and HP.

FILES

/etc/default/fs	Specifies the default file system type
/etc/fstab	Static information about the file systems
/etc/mnttab	Mounted file system table
/etc/passwd	Password file (contains user names)

SEE ALSO

quot_hfs(1M), quot_vxfs(1M), du(1), find(1), ls(1), fstyp(1M), mount(1M), ncheck(1M), repquota(1M), fs_wrapper(5), quota(5).

NAME

quot - summarize ownership on an HFS file system

SYNOPSIS

/usr/sbin/quot [**-F hfs**] [**-V**] [**-cfhnv**] *filesystem* ...

/usr/sbin/quot [**-F hfs**] [**-V**] [**-cfhnv**] **-a**

DESCRIPTION

The **quot** command displays the number of 1024-byte blocks in the named HFS *filesystem* that are currently owned by each user. *filesystem* is either the name of the directory on which the file system is mounted or the name of the device containing the file system.

Options

quot recognizes the following options:

-F hfs Specify the file system type **hfs**.

-V Echo the completed command line, but perform no other action. The command line is generated by incorporating the user-specified options and other information derived from **/etc/fstab.** This option allows the user to verify the command line. If the options specified are valid, the completed command line is echoed. If the options specified are not valid, an error message is printed.

-a Generate a report for all mounted HFS file systems.

-c Report size rather than user statistics. Generates histogram statistics in 3-column format:

Column 1: File size in blocks. Sizes are listed in ascending order up to 499 blocks per file. Files occupying 499 or more blocks are counted together on a single line as 499-block files (but column 3 is based on actual number of blocks occupied).

Column 2: Number of files of size indicated in column 1.

Column 3: Cumulative total blocks occupied by files counted in current plus all preceding lines.

Use of this option overrides the **-f** and **-v** options.

-f Display number of files and space occupied by each user.

-h Calculate the number of blocks in the file based on file size rather than actual blocks allocated. This option does not account for sparse files (files with holes in them).

-n Accept data from the **ncheck** command (see *ncheck*(1M)) as input. Run the pipeline:

ncheck *device* **| sort +0n | quot -n** *filesystem*

to produce a list of all files and their owners.

-v Display three columns containing the number of blocks not accessed in the last 30, 60, and 90 days.

AUTHOR

quot, a disk quota command, was developed by the University of California, Berkeley, Sun Microsystems, Inc., and HP.

FILES

/etc/fstab	Static information about the file systems
/etc/mnttab	Mounted file system table
/etc/passwd	Password file (contains user names).

SEE ALSO

quot(1M), du(1), find(1), ls(1), fstyp(1M), mount(1M), ncheck(1M), repquota(1M), quota(5).

NAME

quot - summarize ownership on a VxFS file system

SYNOPSIS

/usr/sbin/quot [**-F vxfs**] [**-cfhnv**] [**-V**] *filesystem* ...

/usr/sbin/quot [**-F vxfs**] [**-cfhnv**] [**-V**] **-a**

DESCRIPTION

quot displays the number of 1024-byte blocks in the specified VxFS *filesystem* that are currently owned by each user. *filesystem* is either the name of the directory on which the file system is mounted or the name of the device containing the file system.

Options

quot recognizes the following options:

-F vxfs
: Specifies file system type **vxfs**

-a Generate a report for all mounted file systems.

-c Report file system size instead of user statistics. The **-c** option generates histogram statistics in 3-column format:

Column 1:	File size in blocks. Sizes are listed in ascending order up to 499 blocks per file. Files occupying 499 or more blocks are counted together on a single line as 499-block files, but column 3 is based on actual number of blocks occupied.
Column 2:	Number of files of size indicated in column 1.
Column 3:	Cumulative total blocks occupied by files counted in current plus all preceding lines.

The **-c** option overrides the **-f** and **-v** options.

-f Display the number of files and space occupied by each user.

-h Calculate the number of blocks in the file based on file size rather than actual blocks allocated. The **-h** option does not account for sparse files (files with holes in them).

-n Accept *ncheck*(1M) data as input. The following pipeline will produce a list of all files and their owners:

ncheck *device* **| sort +0n | quot -n** *filesystem*

-v Display three columns containing the number of blocks not accessed in the last 30, 60, and 90 days.

-V Validate the command line options, but do not execute the command. If the options specified are valid, **quot -V** echoes the complete command line. If the options specified are not valid, it prints an error message.

EXAMPLES

The following examples show the output of **quot** using various options.

```
# quot /tstmnt

/dev/dsk/c0t1d0 (/tstmnt):
109 auser
 10 root

# quot -v /tstmnt

/dev/dsk/c0t1d0 (/tstmnt):
109 auser 0 0 0
 10 root  0 0 0

# quot -c /tstmnt

/dev/dsk/c0t1d0 (/tstmnt1):
0 2 0
1 4 4
```

```
2 4 12
3 5 27
4 1 31
7 1 38
10 2 58
12 1 70
15 1 85
16 1 101
18 1 119
2047 0 119
```

AUTHOR

Disk Quotas were developed by the University of California, Berkeley, Sun Microsystems, Inc., and HP.

FILES

`/etc/mnttab`	Mounted file system table
`/etc/passwd`	Password file (contains user names).

SEE ALSO

quot(1M), du(1), find(1), fstyp(1M), ls(1), mount(1M), ncheck(1M), repquota(1M), quota(5).

NAME

quotacheck - file system quota consistency checker

SYNOPSIS

/usr/sbin/quotacheck [**-F** *FStype*] [**-V**] [**-o** *specific-options*] *filesystem* ...

/usr/sbin/quotacheck [**-F** *FStype*] [**-V**] [**-o** *specific-options*] **-a**

DESCRIPTION

The **quotacheck** command examines each file system, builds a table of current disk usage, and compares this table against that stored in the disk quota file for the file system. If any inconsistencies are detected, both the quota file and the current system copy of the incorrect quotas are updated.

quotacheck expects each file system to be checked to have a file named **quotas** in the root directory. If none is present, **quotacheck** reports an error and ignores the file system. **quotacheck** is normally run at mount time from start-up scripts.

filesystem represents a mount point or block special device such as **/dev/dsk/c1t0d2**.

Options

quotacheck recognizes the following options:

-F *Fstype* Specify the file system type on which to operate (see *fstyp*(1M) and *fs_wrapper*(5)). If this option is not included on the command line, then the file system type is determined from the file **/etc/fstab** by matching *filesystem* with an entry in that file. If there is no entry in **/etc/fstab**, then the file system type is determined from the file **/etc/default/fs**.

-V Echo the completed command line, but perform no other action. The command line is generated by incorporating the user-specified options and other information derived from **/etc/fstab**. This option allows the user to verify the command line.

-o *specific-options*
Specify options specific to each file system type. *specific-options* is a list of suboptions and/or keyword/attribute pairs intended for a *FStype*-specific module of the command. See the file system specific man pages for a description of the *specific-options* supported, if any.

-a Obtain list of file systems to check from **/etc/fstab**. Only mounted **rw** (or **default**) type file systems with the **quota** option are checked.

q

EXTERNAL INFLUENCES

Environment Variables

LC_MESSAGES determines the language in which messages are displayed.

If **LC_MESSAGES** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**.

If any internationalization variable contains an invalid setting, **quotacheck** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported.

AUTHOR

quotacheck was developed by HP and the University of California, Berkeley.

FILES

/etc/default/fs	Specifies the default file system type
/etc/fstab	Default list of file systems to check
/etc/mnttab	Mounted file system table
*directory***/quotas**	Quota statistics static storage for file system where *directory* is the file system root as specified to the **mount** command (see *mount*(1M)).

SEE ALSO

fs_wrapper(5), mount(1M), quota(5), quotacheck_hfs(1M), quotacheck_vxfs(1M).

NAME

quotacheck - quota consistency checker for HFS file systems

SYNOPSIS

/usr/sbin/quotacheck [**-F hfs**] [**-V**] [**-pPv**] *filesystem* ...

/usr/sbin/quotacheck [**-F hfs**] [**-V**] [**-pPv**] **-a**

DESCRIPTION

The **quotacheck** command examines each HFS file system, builds a table of current disk usage, and compares this table against that stored in the disk quota file for the file system. If any inconsistencies are detected, both the quota file and the current system copy of the incorrect quotas are updated.

quotacheck expects each file system to be checked to have a file named **quotas** in the root directory. If none is present, **quotacheck** reports an error and ignores the file system. **quotacheck** is normally run at mount time from start up scripts.

filesystem represents a mount point or block special device (e.g., **/dev/dsk/c1t0d2**).

Options

quotacheck recognizes the following options:

-F hfs	Specify the file system type **hfs**.
-V	Echo the completed command line, but perform no other action. The command line is generated by incorporating the user-specified options and other information derived from **/etc/fstab.** This option allows the user to verify the command line.
-a	Obtain list of file systems to check from **/etc/fstab**. Only mounted file systems of type **hfs** and **rw** (or **default**) with the **quota** option are checked.
-v	Indicate the calculated disk quotas for each user on a particular file system. **quotacheck** normally reports only those quotas that are modified.
-p	Check file systems in parallel as allowed by equal values in the *pass number* field in **/etc/fstab**.
-P	Preen file systems, checking only those with invalid quota statistics (**quotaoff** and **edquota** commands can invalidate quota statistics as discussed in *quota*(5) — see *quotaoff*(1M) and *edquota*(1M)). Also checks in parallel as in **-p** above.

AUTHOR

quotacheck was developed by HP and the University of California, Berkeley.

q

FILES

/etc/fstab	Static information about the file systems
/etc/mnttab	Mounted file system table
*directory***/quotas**	Quota statistics static storage for filesystem where *directory* is the file system root as specified to the **mount** command (see *mount*(1M)).

SEE ALSO

mount(1M), quota(5), quotacheck(1M), quotaon(1M), quotaoff(1M).

NAME

quotacheck - VxFS file system quota consistency checker

SYNOPSIS

/usr/sbin/quotacheck [**-F vxfs**] [**-V**] [**-pPv**] *filesystem*...

/usr/sbin/quotacheck [**-F vxfs**] [**-V**] [**-pPv**] **-a**

DESCRIPTION

Because VxFS maintains quota information in the kernel, **quotacheck** for VxFS synchronizes quotas from the current system copy to the disk quota file for the specified VxFS file system.

quotacheck requires that each file system it checks has a file named **quotas** in the root directory. **quotacheck** is typically run at mount time from a start-up script.

filesystem is a mount point or block special device (e.g., **/dev/dsk/c0t0d0)**.

Options

quotacheck recognizes the following options:

-a Check the file systems listed in the **/etc/fstab** file. Checks only mounted **rw** type file systems with the **quota** option.

-F vxfs
Specify the file-system type **vxfs**.

-p This option does nothing, but exists for standards compatibility.

-P This option does nothing, but exists for standards compatibility.

-V Echo the completed command line, but do not execute the command. The command line is generated by incorporating the user-specified options and other information derived from **/etc/fstab**. This option allows the user to verify the command line.

-v Report the file system name before synchronizing quotas from current system copy to the disk quota file.

AUTHOR

quotacheck was developed by HP and the University of California, Berkeley.

FILES

/etc/fstab	Default file systems
directory/**quotas**	Quota statistics static storage for file system where *directory* is the file system root as specified to **mount** (see *mount*(1M)).

SEE ALSO

quota(5), quotacheck(1M).

q

NAME

quotaon, quotaoff - turn HFS file system quotas on and off

SYNOPSIS

/usr/sbin/quotaon [**-v**] *filesystem* ...

/usr/sbin/quotaon [**-v**] **-a**

/usr/sbin/quotaoff [**-v**] *filesystem* ...

/usr/sbin/quotaoff [**-v**] **-a**

Remarks

These commands are provided for compatibility only. Their use is neither required nor recommended because **mount** and **umount** enable and disable quotas cleanly (see *mount*(1M)). See *WARNINGS* below for more information.

DESCRIPTION

The **quotaon** command enables quotas on one or more HFS file systems.

The **quotaoff** command disables quotas on one or more HFS file systems.

filesystem is either the name of the mount point of the file system, or the name of the block device containing the file system. The file systems specified must be currently mounted in order to turn quotas on or off. Also, the file system quota file, **quotas**, must be present in the root directory of each specified file system.

These commands will update the appropriate entries in **/etc/mnttab** to indicate that quotas are on or off for each file system.

When enabling quotas interactively after boot time, the **quotacheck** command should be run immediately afterward (see *WARNINGS* below).

Use **mount** (see *mount*(1M)) to determine whether quotas are enabled on mounted file systems.

Options

The following options affect the behavior described above.

- **-a** Obtain the *filesystem* list from **/etc/fstab**, using entries of type **hfs** and **rw** (or **default**) with the **quota** option (see *fstab*(4)).
- **-v** Generate a message for each file system affected.

q

EXTERNAL INFLUENCES

Environment Variables

LC_MESSAGES determines the language in which messages are displayed.

If **LC_MESSAGES** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**.

If any internationalization variable contains an invalid setting, **quotaon** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported.

WARNINGS

Using **quotaoff** to disable quotas on a file system causes the system to discontinue tracking quotas for that file system, and marks the **quota clean** flag in the superblock **NOT_OK** (see *fsclean*(1M)). This in turn, forces a **quotacheck** the next time the system is booted. Since quotas are enabled and disabled cleanly by **mount** and **umount** anyway, the use of **quotaon** and **quotaoff** is generally discouraged.

AUTHOR

Disk quotas were developed by the University of California, Berkeley, Sun Microsystems, Inc., and HP.

FILES

/etc/fstab	Static information about the file systems

`/etc/mnttab`	Mount file system table
*directory***`/quotas`**	Quota statistics storage for the file system, where *directory* is the root of the file system as specified to the **`mount`** command (see *mount*(1M)).

SEE ALSO

fsclean(1M), quotacheck(1M), quotacheck_hfs(1M), quotacheck_vxfs(1M), mount(1M), quota(5).

NAME

rad - perform OLA/R functions without any comprehensive checks

SYNOPSIS

/usr/bin/rad -q

/usr/bin/rad -n

/usr/bin/rad -f *flag slot_id*

/usr/bin/rad -a|c|d|h|i|o|r|s *slot_id*

/usr/bin/rad -C|D|N|R|S|V *hw_path*

DESCRIPTION

The **rad** command provides the ability to perform on-line addition and replacement of I/O cards. This command should be used while the system is in single-user state, when SAM is unavailable.

This tool provides OLA/R capabilities, but is not as comprehensive as the SAM implementation. The command does not make extensive environmental or critical resource checks since it is expected to be executed when the system is in a minimal operating state with few, if any, applications running.

This command is an administrative command and therefore will only allow root privileged users to perform the OLA/R functions.

Options

slot_id	Slot ID of an OLA/R capable slot. A slot ID is a list of one or more numbers separated by dashes. Each number represents a component of the physical location of the slot. The user can use the slot ID to locate the slot. The sequence of numbers in the slot ID is platform dependent. On the N and L classes, the slot ID contains only the slot number. On all other platforms, including SuperDome, the format of the slot ID is: *Cabinet#-Bay#-Chassis#-Slot#*
slot_hw_path	Hardware path of an OLA/R capable slot
device_hw_path	Hardware path of a device as reported by **ioscan**. A single *slot_hw_path* can be a part of multiple *device_hw_path*s for a multi-function or a multi-port card.
-a *slot_id*	List the affected (same power-domain) slot IDs of the specified slot.
-C *device_hw_path*	Display the device information (Device_ID, Vendor_ID, Revision_ID, etc.) of the device at the indicated hardware path.
-c *slot_id*	Display the device information (Device_ID, Vendor_ID, Revision_ID, etc.) of all the interface devices at the indicated slot.
-d *slot_id*	Reserved for future use.
-D *device_hw_path*	Reserved for future use.
-f *flag slot_id*	Set the attention indicator LED of the specified slot. The flag must be one of the following values: **ON**, **ATTENTION**, or **OFF**.
-h *slot_id*	List the hardware paths of the interface node(s) for the specified slot.
-i *slot_id*	Power ON the indicated slot. The other slots, if any, that are on the same power-domain will have their power turned ON as well (potentially harmful to the system operations).
-n	Display the number of OLA/R capable slots in the system.
-N *slot_hw_path* | *device_hw_path*	Find the slot ID of the specified slot or device hardware path.
-o *slot_id*	Power OFF the indicated slot. The other slots, if any, that are on the same power-domain will have their power turned OFF as well (potentially harmful to the system operations).
-q	Display the status of all OLA/R slots on the system.
-r *slot_id*	Resume the operations of the indicated slot and its affected slots (same power-domain).

-R *device_hw_path*	Resume the instance of the driver at the specified hardware path.
-s *slot_id*	Suspend the operations of the indicated slot and its affected slots (same power-domain) (potentially harmful to the system operations).
-S *device_hw_path*	Suspend the instance of the driver at the specified hardware path.
-V *device_hw_path*	Display the OLA/R driver information (current state, timeout values in seconds, etc.) for the driver at the specified hardware path.

Note

In some cases, administrators must initiate certain actions before hardware I/O controller cards are replaced or deleted; other cases require activity after cards are replaced or added. Pre- and Post-OLAR scripts will be used to perform these necessary activities. These scripts, named according to the driver name (if the driver is **foodriver** then the script is **foodriver**), are located in **/usr/sbin/olrad.d** directory. These scripts has the following synopsis:

/usr/sbin/olrad.d/*driver action hardware_path*

action is one of:

post_add	Execute post add actions.
pref_replace	Execute preface to replace actions.
prep_replace	Execute prepare to replace actions.
post_replace	Execute post replace actions.
pref_delete	Execute preface to delete actions.
prep_delete	Execute prepare to delete actions.
post_delete	Execute post delete actions.

hardware_path is the hardware path of the interface node.

For a detailed description of what each script does, please view the individual scripts.

EXAMPLES

Adding a new card

The command invocation sequence for adding a new card:

1. Get information about all the OLA/R capable slots. Make note of the *slot_id* field.

   /usr/bin/rad -q

2. Set the attention indicator LED of the desired slot to make sure that the correct *slot_id* was used.

   /usr/bin/rad -f ATTENTION *slot_id*

3. Get the affected *slot_id*s for this slot; that is, check to see if other slots will be powered down with this slot. If other slots are affected by this slot and if these slots are occupied, then you must suspend them by using the **-s** option to the **rad** command.

   /usr/bin/rad -a *slot_id*

4. Turn off the power to the slot. Note that the power to the other slots in the same power-domain (if any), will be turned off as well.

   /usr/bin/rad -o *slot_id*

5. Insert the new card into the slot.

6. Turn the power on to the slot.

   /usr/bin/rad -i *slot_id*

7. Turn the attention indicator LED off.

   /usr/bin/rad -f OFF *slot_id*

8. Run the **/usr/sbin/ioscan** command with the appropriate options to configure the new card.

Replacing a card

The command invocation sequence for replacing a card:

1. Get information about all the OLA/R capable slots. Make note of the *slot_id* field.

r

/usr/bin/rad -q

2. Set the attention indicator LED of the desired slot to make sure that the correct *slot_id* was used.

   /usr/bin/rad -f ATTENTION *slot_id*

3. Get the affected *slot_id*s for this slot; that is, check to see if other slots will be powered down with this slot.

   /usr/bin/rad -a *slot_id*

4. Suspend the software drivers of this slot as well as the other slots that are affected (**-a** option) by this slot.

   /usr/bin/rad -s *slot_id*

5. Turn off the power to this slot as well as the other slots that are affected (**-a** option) by this slot.

   /usr/bin/rad -o *slot_id*

6. Replace the faulty card in the slot with a working card. The new card must be identical (same HP product number) as the card being replaced.

7. Turn the power on to this slot as well as the other slots that are affected (**-a** option) by this slot.

   /usr/bin/rad -i *slot_id*

8. Resume the software drivers of this slot as well as the other slots that are affected (**-a** option) by this slot.

   /usr/bin/rad -r *slot_id*

9. Turn the attention indicator LED off.

   /usr/bin/rad -f OFF *slot_id*

RETURN VALUE

rad returns 0 on success. On failure, **rad** returns -1 and prints an appropriate error message to standard error.

SEE ALSO

ioscan(1M), sam(1M).

NAME

rarpc - Reverse Address Resolution Protocol client

SYNOPSIS

rarpc [**-d**] [**-e**|**-s**] [**-n** *count*] *interface_name*

DESCRIPTION

rarpc, the Reverse Address Resolution Protocol client, implements the client portion of the Reverse Address Resolution Protocol (see *SEE ALSO*). It sends RARP requests for the specified interface's hardware address and waits for the response from the RARP server. **rarpc** can be used during boot-time initialization to find the IP address of an interface. To do so, set the **IP_ADDRESS**[*i*] variable of interface *i* with **IP_ADDRESS[** *i* **]=RARP** in **/etc/rc.config.d/netconf**.

Options are:

-d	Print debugging information.
-e	Use ethernet encapsulation only.
-s	Use SNAP encapsulation only.
-n *count*	Transmits count requests and waits for each one to time out before giving up.
interface_name	Identifies the interface to request information about.

If a response is received, it prints the IP address to its standard output. This information can be used to configure the interface as seen in **/sbin/init.d/net**.

If a response is not received, the client will retransmit after 2 seconds, and then after 4 seconds. After that, retransmissions occur every 8 seconds.

RETURN VALUE

Exit status is 1 if the command fails or no RARP response is received. Exit status is 0 and the IP address is printed to standard output if a response is received.

LIMITATIONS

1. The **rarpc** client cannot be run at the same time a **rarpd** daemon is running on the same interface.
2. The **rarpc** client supports only ethernet, 100VG and FDDI network interfaces.

AUTHOR

rarpc was developed by HP.

SEE ALSO

rarpd(1M).

R. Finlayson, T. Mann, J.C. Mogul, M. Theimer, "Reverse Address Resolution Protocol", RFC 903.

NAME

rarpd - Reverse Address Resolution Protocol daemon

SYNOPSIS

rarpd [**-d**] [**-f** *config_file*] [*interface_name*]

DESCRIPTION

rarpd, the Reverse Address Resolution Protocol daemon, implements the server portion of the Reverse Address Resolution Protocol [1]. It responds to RARP requests providing the requested client IP address. Rarpd can be started during boot-time initialization. To do so, set the RARPD variable with **RARPD=1** in **/etc/rc.config.d/netconf**.

Options are:

-d	Print debugging information.
-f *config_file*	Use the specified config_file database instead of **/etc/rarpd.conf**.
interface_name	Respond to requests over just this interface.

The configuration file database contains hardware address to IP address mappings. Other than comment lines (which begin with a '#') and blank lines, all lines are considered client entries. A client entry is of the form:

hardware_address WHITE_SPACE *ip_address*

where *hardware_address* consists of (**:**) colon-separated hexadecimal bytes, and *ip_address* consists of (**.**) dot-seperated decimal bytes. For example:

```
#
# hardware addr   IP addr
#
#   ethernet clients
08:00:09:26:ec:19 15.13.136.68
08:00:09:17:0a:93 15.13.136.74
#
#   100VG clients
08:00:09:63:5d:f5 190.20.30.103
#
#   FDDI clients
08:00:09:09:53:4c 192.20.30.98
```

There must be exactly 6 hardware address bytes. There must be exactly 4 protocol address bytes.

The following signals have the specified effect when sent to the rarpd process using the *kill*(1) command:

SIGHUP	Causes server to read the config file and reload database.
SIGINT	Dumps current data base and cache to **/var/tmp/rarpd.db**.

RETURN VALUE

Exit status is 1 if the command fails, and error messages are written to stderr and/or syslog. Typically, the daemon will continue answering requests until externally interrupted.

LIMITATIONS

1. The rarpd daemon supports only ethernet, 100VG and FDDI network interfaces.
2. The rarpd daemon supports only 4 byte Internet Protocol addresses.
3. The rarpd and rarpc programs cannot be run on the same interface at the same time.

AUTHOR

rarpd was developed by HP.

SEE ALSO

rarpc(1M).

[1] R. Finlayson, T. Mann, J.C. Mogul, M. Theimer, "Reverse Address Resolution Protocol", RFC 903.

NAME

rbootd - remote boot server for RMP clients

SYNOPSIS

/usr/sbin/rbootd [**-a**] [**-l** *loglevel*] [**-L** *logfile*] [**-t** *minutes*] [*landevs*]

DESCRIPTION

rbootd services initial boot-up requests from RMP clients over a local area network. Early s700 workstations and all Datacommunications and Terminal Controllers (DTC/9000) use this RMP protocol and can only communicate with **rbootd** during boot-up. Later s700 workstations (starting with the s712) use the industry standard BOOTP protocol and communicate with *bootpd*(1M). Future s700 workstations will use the BOOTP protocol. See the listings below.

rbootd now acts as a forwarding agent for s700 RMP clients, receiving their RMP boot requests and reformulating them into BOOTP boot requests that are sent to the local **bootpd** daemon. If **bootpd** replies to this boot request, **rbootd** receives the BOOTP reply and produces an RMP reply which is sent to the client. **rbootd** continues to act as the intermediary in this transaction until the client is successfully booted.

rbootd only responds to DTC clients if they are listed in the **map802** file. The **map802** file (a binary file) is created when a DTC is configured by *dtcconfig*(1M) on the host machine.

In order to boot a s700 RMP client run **rbootd** and **bootpd** on the server machine, on the same subnet as the client. If the local **bootpd** daemon is acting as a relay agent, there must also be a remote NFS Diskless server with the necessary boot files and NFS or **tftp** access to those files.

Options

rbootd supports the following options:

-a Append to the **rbootd** log file. By default, starting up **rbootd** truncates the log file.

-l *loglevel* Set the amount of information that will be logged in the log file. **rbootd** supports the following logging levels:

0	Log only **rbootd** startup and termination messages.
1	Log all errors. This is the default logging level.
2	Log rejected boot requests from machines not found in **/etc/bootptab** or **/etc/opt/dtcmgr/map802**.
3	Log all boot requests.

-L *logfile* Specify an alternate file that **rbootd** should use to log status and error messages.

-t *minutes* Grace period before removing inactive temporary files. Meaningful only in the **tftp**-remote configuration. Default is 10 minutes.

landevs Specify the *only* devices that **rbootd** should use to listen for boot requests. The default is all LAN devices. The device names must be of the form **lan0** or **lan1** etc, where the device name matches what is reported by **lanscan**

New Functionality

Beginning with HP-UX 10.0 **rbootd** has the following behavior:

- **bootpd/bootptab Dependency :**

  rbootd now relies on *bootpd*(1M) to verify the identity of cluster clients and locate the bootable images (from **/etc/bootptab**). RMP clients are thus administered in exactly the same way as new BOOTP clients. The old methods for administering RMP clients (**/etc/clusterconf**, context-dependent files, **/usr/boot/***) are obsolete and no longer work.

  See *bootpd*(1M) and *sam*(1M) for details on configuring cluster clients.

  It is necessary to have the **bootpd** daemon running on the same machine as the **rbootd** daemon.

- **Auto-Discovery**:

  To aid the system administrator, **rbootd** now discovers working ethernet interfaces at startup time and monitors them for boot requests. Alternatively, the system administrator may put a list of up to ten ethernet devices on the command line. Putting device names on the command line means "monitor these devices ONLY". If device names are included on the command line, they must be ethernet interfaces (not X.25, token-ring, etc) and they must be up and running at the time **rbootd** is started. See *lanscan*(1M) and *ifconfig*(1M) to determine the state of system devices. Attempting to have **rbootd** monitor non-ethernet devices will not succeed. The device names must always be of the form **lan0** or **lan1** etc, where the device name matches what is reported by **lanscan**.

- **Multiple LAN Coverage :**

  rbootd can monitor up to 10 lan devices (depending on hardware) and can boot clients from all of them. Clients are still restricted to booting from their own builtin lan devices.

- **Gateway Booting :**

  RMP clients can now be booted from servers that are not on the same subnet as the client. The RMP boot requests and replies cannot cross gateways, but the repackaged BOOTP requests and replies can. The BOOTP requests and replies are relayed across gateways by **bootpd**. This is known as the remote configuration.

  rbootd uses the NFS or **tftp** mechanism to transfer the necessary files from the remote server to the **rbootd** machine, and then transfers the bootable images to the client in a succession of RMP packets. Thus the remote server must make the necessary files accessible by NFS or **tftp**.

  In the remote-**tftp** case, the boot files are temporarily stored in **/var/rbootd/C0809***, and are removed after a period of inactivity, controlled by the **-t** option. The default is 10 minutes.

- **S800 Servers :**

  S800 machines can now be used as cluster servers, booting s700 clients and DTCs. S800 machines are not supported as cluster clients.

- **Network Install :**

  rbootd now forwards install requests to *instl_bootd*(1M). If there is no appropriate response, **rbootd** will deny the request.

- **S300/400 Not Supported :**

  S300/400 machines are not supported as diskless clients.

- **Performance Recommendations :**

  Boot from a local server for the fastest boot times. Run the **rbootd** daemon and the **bootpd** server daemon on the same machine, and avoid transferring the boot files by NFS or **tftp**. This is strongly recommended.

  If booting from remote **bootpd** servers (across gateways), use NFS mounts to make the boot files available to the **rbootd** server. See *mount*(1M) for more information. The system administrator can configure local and remote diskless clients in any mix, but it is strongly recommended that the number of remote diskless clients be minimized.

  If booting from remote servers using the **tftp** method, there must also be temporary file space available on the **rbootd** server machine. Generally 6-8 MBytes per diskless client must be available under **/var**, but this number could be larger when booting customized kernels. These temporary files are removed automatically after some period of inactivity, controlled by the **-t** option. The default is 10 minutes.

- **RMP/BOOTP :**

  The RMP clients are the older s700 workstations and all DTCs: workstations: 705, 710, 715/33, 715/50, 715/75, 720, 725/50, 725/75, 730, 735, 750, 755

  The BOOTP clients are the s712, s715/64, s715/100, B-Class, C-Class, D-Class and future workstations.

WARNINGS

It is necessary to stop **rbootd** before running **bootpquery** because they use the same reserved port (67/udp).

The **rbootd** daemon binds to port 1067 for cold-install clients through **instl_bootd**. Because this is not a reserved port, sometimes **rbootd** will be unable to start when another process is holding this port. Use **netstat -an** to find the other process and kill it. Rebooting is also an option.

AUTHOR

rbootd was developed by HP.

FILES

`/var/adm/rbootd.log`	Default rbootd log file.
`/etc/boottab`	Bootstrap configuration file.
`/etc/opt/dtcmgr/map802`	DTC/9000 configuration file.
`/var/rbootd/C0809*`	Temporary boot files.

Obsoleted Files

`/etc/clusterconf`
`/usr/boot/*`

SEE ALSO

bootpd(1M), instl_bootd(1M), tftpd(1M), mount(1M), sam(1M), dcnodes(1), dtcconfig(1M), dtcnmd(1M), dtcnmp(1M).

r

NAME

rc - general purpose sequencer invoked upon entering new run level

SYNOPSIS

/sbin/rc

DESCRIPTION

The **rc** shell script is the general sequencer invoked upon entering a new run level via the **init** *N* command (where *N* equals 0-6). The script **/sbin/rc** is typically invoked by the corresponding entry in the file **/etc/inittab** as follows:

```
sqnc:123456:wait:/sbin/rc </dev/console >/dev/console 2>&1
```

/sbin/rc is the startup and shutdown sequencer script. There is only one sequencer script and it handles all of the sequencer directories. This script sequences the scripts in the appropriate sequencer directories in alphabetical order as defined by the shell and invokes them as either startup or kill scripts.

If a transition from a lower to a higher run level (i.e., *init* state) occurs, the start scripts for the new run level and all intermediate levels between the old and new level are executed. If a transition from a higher to a lower run level occurs, the kill scripts for the new run level and all intermediate levels between the old and new level are executed.

If a start script link (e.g., **/sbin/rc***N***.d/S123test**) in sequencer *N* has a stop action, the corresponding kill script should be placed in sequencer *N***-1** (e.g., **/sbin/rc***N***-1.d/K200test**). Actions started in level *N* should be stopped in level *N***-1**. This way, a system shutdown (e.g., transition from level 3 directly to level 0) will result in all subsystems being stopped.

Start and Kill Scripts

In many cases, a startup script will have both a start and a kill action. For example, the *inetd* script starts the Internet daemon in the start case, and kills that process in the stop case. Instead of two separate scripts, only one exists, which accepts both the **start** and **stop** arguments and executes the correct code. In some cases, only a start action will be applicable. If this is the case, and if the **stop** action is specified, the script should produce a usage message and exit with an error. In general, scripts should look at their arguments and produce error messages if bad arguments are present. When a script executes properly, it must exit with a return value of zero. If an error condition exists, the return value must be nonzero.

Naming Conventions

The startup and shutdown scripts (referred to as startup scripts hereafter) exist in the **/sbin/init.d** directory, named after the subsystem they control. For example, the **/sbin/init.d/cron** script controls starting up the **cron** daemon. The contents of sequencer directories consist of symbolic links to startup scripts in **/sbin/init.d**. These symbolic links must follow a strict naming convention, as noted in the various fields of this example:

```
/sbin/rc2.d/S060cron
```

where the fields are defined as follows:

rc2.d The sequencer directory is numbered to reflect the run level for which its contents will be executed. In this case, start scripts in this directory will be executed upon entering run level 2 from run level 1, and kill scripts will be executed upon entering run level 2 from run level 3.

S The first character of a sequencer link name determines whether the script is executed as a start script (if the character is **S**), or as a kill script (if the character is **K**).

060 A three digit number is used for sequencing scripts within the sequencer directory. Scripts are executed by type (start or kill) in alphabetical order as defined by the shell. Although it is not recommended, two scripts may share the same sequence number.

cron The name of the startup script follows the sequence number. The startup script name must be the same name as the script to which this sequencer entry is linked. In this example, the link points to **/sbin/init.d/cron**.

Note that short file name systems require file names of 14 or less characters. This means that the fourth field is limited to 10 or fewer characters.

Scripts are executed in alphabetical order. The entire file name of the script is used for alphabetical ordering purposes.

When ordering start and kill script links, note that subsystems started in any given order should be stopped in the reverse order to eliminate any dependencies between subsystems. This means that kill scripts will generally not have the same numbers as their start script counterparts. For example, if two subsystems must be started in a given order due to dependencies (e.g., **S111house** followed by **S222uses_house**), the kill counterparts to these scripts must be numbered so that the subsystems are stopped in the opposite order in which they were started (e.g., **K555uses_house** followed by **K777house**).

Also keep in mind that kill scripts for a start script in directory **/sbin/rc***N***.d** will reside in **/sbin/rc(***N***-1).d**. For example, **/sbin/rc3.d/S123homer** and **/sbin/rc2.d/K654homer** might be start/kill counterparts.

Arguments

The startup/shutdown scripts should be able to recognize the following four arguments (where applicable):

start The **start** argument is passed to scripts whose names start with **S**. Upon receiving the **start** argument, the script should perform its start actions.

stop The **stop** argument is passed to scripts whose names start with **K**. Upon receiving the **stop** argument, the script should perform its stop actions.

start_msg The **start_msg** argument is passed to scripts whose names start with **S** so that the script can report back a short message indicating what the start action will do. For instance, when the **lp** spooler script is invoked with a **start_msg** argument, it echoes

```
Starting the LP subsystem
```

This string is used by the startup routines. Scripts given just the **start_msg** argument will only print a message and not perform any actions.

stop_msg The **stop_msg** argument is passed to scripts whose names start with **K** so that the script can report back a short message indicating what the stop action will do. For instance, when the **lp** spooler script is invoked with a **stop_msg** argument, it echoes

```
Stopping the LP subsystem
```

This string is used by the shutdown checklist. Scripts given just the **stop_msg** argument will only print a message and not perform any actions.

Script Output

To ensure proper reporting of startup events, startup scripts are required to comply with the following guidelines for script output.

- Status messages, such as

  ```
  starting house daemon
  ```

  must be directed to stdout. All error messages must be directed to stderr.

- Script output, both stdout and stderr, is redirected to log file **/etc/rc.log**, unless the startup checklist mode is set to the raw mode. In this case, all output goes to the console. All error messages should be echoed to stdout or stderr.

- Startup scripts are not allowed to send messages directly to the console, or to start any daemons that immediately write to the console. This restriction exists because these scripts are now started by the **/sbin/rc** checklist wrapper. All script output should go to either stdout or stderr, and thus be captured in a log file. Any console output will be garbled.

RETURN VALUE

The return values for startup scripts are as follows:

0 Script exited without error.
1 Script encountered errors.
2 Script was skipped due to overriding control variables from **/etc/rc.config.d** files, or for other reasons, and did not actually do anything.
3 Script will automatically reboot the system.

4	Script exited without error and started a process in background mode.
>4	For return values greater than **4** the action is same as return value **1**, script encountered errors.

SEE ALSO

init(1M), shutdown(1M), inittab(4), rc.config(4).

NAME

rcancel - remove requests from a remote line printer spooling queue

SYNOPSIS

/usr/sbin/rcancel [*id* ...] [*printer*] [**-a**] [**-e**] [**-u** *user*]

DESCRIPTION

The **rcancel** command removes a request, or requests, from the spool queue of a remote printer. **rcancel** is invoked by the **cancel** command (see *cancel*(1)).

At least one *id* or the name of a *printer* must be specified.

This command is intended to be used only by the spool system in response to the **cancel** command (see *lp*(1)), and should not be invoked directly.

Options

The **rcancel** command recognizes the following options:

id	Specifying a request ID (as returned by **lp** (see *lp*(1)) cancels the associated request (if the request is owned by the user), even if it is currently printing.
printer	Name of the printer (for a complete list, use **lpstat** (see *lpstat*(1)). Specifying a *printer* cancels the request which is currently printing on that printer, if the request is owned by the user. If the **-a**, **-e**, or the **-u** option is specified, this option only specifies the printer on which to perform the cancel operation.
-a	Remove all requests owned by the user on the specified printer (see *printer*). The owner is determined by the user's login name and host name on the machine where the **lp** command was invoked.
-e	Empty the spool queue of all requests for the specified *printer*. This form of invoking **rcancel** is useful only to users with appropriate privileges.
-u *user*	Remove any requests queued belonging to that user (or users). This form of invoking **rcancel** is available only to users with appropriate privileges.

AUTHOR

rcancel was developed by the University of California, Berkeley, and HP.

FILES

/var/spool/lp/*
/var/adm/lp/*
/etc/lp/*
/usr/lib/lp/*

SEE ALSO

enable(1), lp(1), lpstat(1), accept(1M), lpadmin(1M), lpsched(1M), rlp(1M), rlpdaemon(1M), rlpstat(1M).

NAME

rdpd - router discovery protocol daemon (OBSOLESCENT)

SYNOPSIS

rdpd [**-r** | **-t** | **-v**]

DESCRIPTION

rdpd, the router discover protocol daemon, implements the host portion of the router discovery protocol (see *SEE ALSO*). More specifically **rdpd**:

- solicits router advertisements when it is first started so as to populate the kernel table as soon as possible.
- listens on all ethernet interfaces (that are up) for ICMP router advertisement datagrams.
- adds a default router to the kernel table based on whether the router is a neighbor and has the highest preference among all advertisements received.
- ages the default router entry applied to the kernel table based on the lifetime value found in the advertisement message.

rdpd can be started during boot-time initialization. To do so, see **/etc/rc.config.d/netconf**. (But see *WARNINGS* below.)

Options

rdpd supports the following options:

-r Display the version of **rdpd**.

-t Enable tracing of the following events:

- setting of expiration timer for advertised entry.
- expiration of a router advertisement entry (only the active entry has a timer running).
- add/update of an advertised router to the kernel.
- removal from kernel table of an advertised router.
- reception of a router advertisement from the link.
- transmission of a router solicitation message.
- failure while attempting to transmit a solicitation.

-v Enable verbose tracing, which in addition to the above, traces:

- contents of the router advertisement message received.
- contents of *rdpd* internal statics which includes:
  1. total number of **icmp** messages received,
  2. total number advertisements received,
  3. total number of advertisements with invalid number of addresses field,
  4. total number of advertisements with invalid address size field,
  5. total number of advertisements with invalid message lengths,
  6. total number of advertisements with invalid lifetime fields,
  7. total number of **icmp** messages with number of bytes received <> header length field.

r

LIMITATIONS

1. The maximum number of default routes retained is 10. Only one of which is applied to the kernel routing tables (the one with the highest preference). In the event that the advertised router with the highest preference expires the retained advertised router list will be searched for the highest preference, still current entry and that entry will be applied to the kernel table. This allows for quick recovery from aged advertisements.
2. **rdpd** only becomes aware of link state changes when either a new Router Advertisement message is received or a timer pops to age a currently active default router added by **rdpd**. This may cause a delay between an interface state change (e.g., **ifconfig** down) and any necessary change to the kernel routing table.
3. The "all hosts on subnet" broadcast address is used for sending solicitations instead of either the all-routers multicast or limited-broadcast IP addresses.
4. The limited-broadcast address for inbound Advertisements is assumed.

5. Default routers added via the **route** command can be altered due to Router Advertisements for the same router.
6. Adding default routes via the **route** command can cause unpredictable results and should be avoided.

OBSOLESCENCE

The functionality of **rdpd** has been subsumed in **gated**. See the **routerdiscovery** statements described in *gated.conf*(4). Consequently, **rdpd** may be obsoleted in a future release of HP-UX.

WARNINGS

rdpd should not be used if **routerdiscovery client** is enabled when running **gated**.

AUTHOR

rdpd was developed by HP.

SEE ALSO

gated(1M), gated.conf(4).

[1] Deering, S., "ICMP Router Discovery Messages", RFC 1256

NAME

reboot - reboot the system

SYNOPSIS

/usr/sbin/reboot [**-h**|**-r**] [**-n**|**-s**] [**-q**] [**-t** *time*] [**-m** *message*]

/usr/sbin/reboot -R [**-H**] [**-n**|**-s**] [**-q**] [**-t** *time*] [**-m** *message*]

DESCRIPTION

The **reboot** command terminates all currently executing processes except those essential to the system, then reboots the system, or halts, or makes the partition ready for reconfig. When invoked without arguments, **reboot** syncs all disks before rebooting the system.

Options

The **reboot** command recognizes the following options:

-h	Shut down the system and halt.
-r	Shut down the system and reboot automatically (default).
-R	Shut down the system to a ready to reconfig state and reboot automatically. This option is available only on systems that support hardware partitions.
-H	Shut down the system to a ready to reconfig state and do not reboot. This option can be used only in combination with the **-R** option. This option is available only on systems that support hardware partitions.
-n	Do not sync the file systems before shutdown.
-s	Sync the file systems before shutdown; for file systems that were cleanly mounted, modify the **fs_clean** flag from **FS_OK** to **FS_CLEAN** (default).
-q	Quick and quiet. Suppress broadcast of warning messages, terminate processes by brute force (with **SIGKILL**) and immediately call **reboot** with arguments as indicated by the other options (see *reboot*(2)). No logging is performed. The **-t** and **-m** options are ignored with this option.
-t *time*	Specify what time **reboot** will bring the system down. *time* can be the word **now** (indicating immediate shutdown) or a future time in one of two formats: **+***number* and *hour***:***min*. The first form brings the system down in *number* minutes; the second brings the system down at the time of day indicated (based on a 24-hour clock).
-m *message*	Display *message* at the terminals of all users on the system at decreasing intervals as reboot *time* approaches. The *message* must not contain any embedded double quotes.

At shutdown time a message is written in the file

/etc/shutdownlog

(if it exists), containing the time of shutdown, who ran **reboot**, and the reason.

Only users with appropriate privileges can execute the **shutdown** command.

WARNINGS

reboot does not invoke the shutdown scripts associated with subsystems to bring them down in a cautious manner. See *shutdown*(1M).

AUTHOR

reboot was developed by HP and the University of California, Berkeley.

FILES

/etc/shutdownlog	Shutdown log

SEE ALSO

reboot(2).

NAME

remshd - remote shell server

SYNOPSIS

/usr/lbin/remshd [**-lns**]

DESCRIPTION

The **remshd** command is the server for the **rcp**, **rdist** and **remsh** commands, and the **rcmd()** function (see *rcp*(1), *rdist*(1), *remsh*(1), and *rcmd*(3N)). The server provides remote execution facilities with authentication based on privileged port numbers.

The **inetd** daemon calls **remshd** when a service request is received at the port indicated for the **shell** (or **cmd**) service specified in **/etc/services** (see *inetd*(1M) and *services*(4)). When called, **inetd** creates a connection to the service on the client's host. To run **remshd**, the following line should be present in the **/etc/inetd.conf** file:

```
shell  stream  tcp  nowait  root  /usr/lbin/remshd  remshd
```

See *inetd.conf*(4) for more information.

Options

remshd recognizes the following options.

-l Disallow authentication based on the user's **.rhosts** file unless the user is a superuser.

-n Disable transport-level keep-alive messages. Otherwise, the messages are enabled. The keep-alive messages allow sessions to be timed out if the client crashes or becomes unreachable.

-s This option is used in multi-homed NIS systems. It disables **remshd** from doing a reverse lookup, of the client's IP address; see *gethostbyname*(3N). It can be used to circumvent an NIS limitation with multihomed hosts.

Operation

When **remshd** receives a service request, it responds with the following protocol:

1. The server checks the client's source port. If the port is not in the range 512 through 1023, the server aborts the connection.
2. The server reads characters from the connection up to a null (**\0**) byte. It interprets the resulting string as an ASCII number, base 10.
3. If the number is non-zero, it is interpreted as the port number of a secondary stream to be used for standard error. A second connection is then created to the specified port on the client's host. (The source port of this second connection must be also in the range 512 through 1023.) If the first character sent is a null (**\0**), no secondary connection is made, and the standard error from the command is sent to the primary stream. If the secondary connection has been made, **remshd** interprets bytes it receives on that socket as signal numbers and passes them to the command as signals. See *signal*(2).
4. The server checks the client's source address and requests the corresponding host name (see *named*(1M), *gethostbyaddr*(3N), and *hosts*(4)). If it cannot determine the hostname, it uses the dot-notation representation of the host address.
5. The server reads the client's host account name from the first connection. This is a null-terminated sequence not exceeding 16 characters.
6. The server reads the server's host account name from the first connection. This is a null-terminated sequence not exceeding 16 characters.
7. The server reads a command to be passed to the shell from the first connection. The command length is limited by the maximum size of the system's argument list.
8. **remshd** then validates the user as follows (all actions take place on the host **remshd** runs on):
   a. It looks up the user account name (retrieved in step 6) in the password file. If it finds it, it performs a **chdir()** to either the user's home directory, if there is one, or to "/."
   b. If either the lookup or **chdir()** fails, the connection is terminated (see *chdir*(2)).
   c. The connection is also terminated if

- the account accessed is administratively locked. The account can be locked by entering a character in the password field that is not part of the set of digits (such as *). The characters used to represent "digits" are . for 0, / for 1, 0 through 9 for 2 through 11, A through Z for 12 through 37, and a through z for 38 through 63. (See also *passwd*(4)).
- the account accessed is protected by a password and, either the password expired or the account on the client's host is not equivalent to the account accessed;
- **remshd** runs on a secure system and the account accessed is not protected by a password.

For more information on equivalent accounts, see *hosts.equiv*(4).

9. A null byte is returned on the primary connection and the command line is passed to the normal login shell of the user with that shell's **-c** option. The shell inherits the network connections established by **remshd** and assumes the normal user and group permissions of the user.

   remshd uses the following path when executing the specified command:

   /usr/bin:/usr/ccs/bin:/usr/bin/X11:/usr/contrib/bin:/usr/local/bin

10. If a secondary socket has been set up, **remshd** normally exits when command standard error and secondary socket standard error have both been closed. If no secondary socket was set up, **remshd** has called an *exec*(2) function, launched the command process, and is no longer present.

DIAGNOSTICS

All diagnostic messages are returned on the connection associated with standard error after which any network connections are closed. An error is indicated by a leading byte with a value of 1 (0 is returned in step 9 above upon successful completion of all the steps before the command execution).

Malformed from address

The first socket connection does not use a reserved port or the client's host address is not an Internet address.

Can't get stderr port

Unable to complete the connection of the secondary socket used for error communication.

Second port not reserved

The secondary socket connection does not use a reserved port.

Locuser too long

The name of the user account on the client's host is longer than 16 characters.

Remuser too long

The name of the user on the server's host is longer than 16 characters.

Command too long

The command line passed exceeds the size of the argument list (as configured into the system).

Login incorrect

No password file entry existed for the user name on the server's host, or the authentication procedure described above in step 8 failed.

No remote directory

The **chdir** command to the home directory or "/" on the server's host failed.

Can't make pipe

The pipe needed for the standard error output wasn't created.

No more processes

The server was unable to fork a process to handle the incoming connection.

Next step: Wait a period of time and try again. If this message persists, the server's host may have runaway processes that are using all the entries in the process table.

system call **:** *message*

Error in executing the named system call. The message specifies the cause of the failure.

shellname: ...

The user's login shell could not be started. This message is returned on the connection associated with the standard error, and is not preceded by a leading byte with a value of 1. Other messages can be returned by the remote command when it executes.

WARNINGS

The "privileged port" authentication procedure used here assumes the integrity of each host and the connecting medium. This is insecure, but is useful in an "open" environment.

remshd ignores **SIGHUP**, **SIGINT**, **SIGQUIT**, and **SIGTERM**, so these signal numbers can safely be sent to remote commands via the secondary socket provided by **remshd**. Other signal numbers may cause **remshd** to kill itself.

AUTHOR

remshd was developed by the University of California, Berkeley.

FILES

$HOME/.rhosts	User's private equivalence list
/etc/hosts.equiv	List of equivalent hosts

SEE ALSO

remsh(1), inetd(1M), named(1M), rcmd(3N), hosts(4), hosts.equiv(4), inetd.conf(4), inetd.sec(4), services(4).

NAME

remshd - remote shell server

SYNOPSIS

/usr/lbin/remshd [**-ln**]

In Kerberos V5 Network Authentication environments:

/usr/lbin/remshd [**-clnKkRr**]

DESCRIPTION

The **remshd** command is the server for the **rcp**, **rdist** and **remsh** commands, and the **rcmd()** function (see *rcp*(1), *rdist*(1), *remsh*(1), and *rcmd*(3N)).

remshd allows two kinds of authentication methods:

1. Authentication based on privileged port numbers where the client's source port must be in the range 512 through 1023. In this case **remshd** assumes it is operating in normal or non-secure environment.
2. Authentication based on Kerberos V5. In this case **remshd** assumes it is operating in a Kerberos V5 Network Authentication, i.e., secure environment.

The *inetd* daemon invokes **remshd** if a service request is received at ports indicated by **shell** or **kshell** services specified in **/etc/services** (see *inetd*(1M) and *services*(4)). Service requests arriving at the **kshell** port assume a secure environment and expect Kerberos authentication to take place.

To start **remshd** from the *inetd* daemon in a non-secure environment, the configuration file **/etc/inetd.conf** must contain an entry as follows:

```
shell  stream  tcp  nowait  root  /usr/lbin/remshd  remshd
```

In a secure environment, **/etc/inetd.conf** must contain an entry:

```
kshell  stream  tcp  nowait  root  /usr/lbin/remshd  remshd -K
```

See *inetd.conf*(4) for more information.

To prevent non-secure access, the entry for **shell** should be commented out in **/etc/inetd.conf**. Any non-Kerberos access will be denied since the entry for the port indicated by **shell** has now been removed or commented out. In a such a situation, a generic error message,

rcmd: connect <hostname> : Connection refused

is displayed. See *DIAGNOSTICS* for more details. Note: by commenting out the entry for the port, access by other clients such as **rdist** will also be prevented.

r

Options

remshd recognizes the following options.

-c Ignore checksum verification. This option is used to achieve interoperability between clients and servers using different checksum calculation methods. For example, the checksum calculation in a application developed with Kerberos V5 Beta 4 API is different from the calculation in a Kerberos V5-1.0 application.

-l Disallow authentication based on the user's **.rhosts** file unless the user is a superuser.

-n Disable transport-level keep-alive messages. Otherwise, the messages are enabled. The keep-alive messages allow sessions to be timed out if the client crashes or becomes unreachable.

In a secure environment, **remshd** will recognize the following additional options:

-K Authorization based on Kerberos V5 must succeed or access will be rejected. (see *sis*(5) for details on authorization).

-R Authentication based on privileged port numbers and authorization of the remote user through equivalent accounts must succeed. For more information on equivalent accounts, see *hosts.equiv(4).*

-r Either one of the following must succeed. The order in which the authorization checks are done is as specified below.

1. Authentication based on privileged port numbers and authorization of the remote user through equivalent accounts (see *hosts.equiv*(4)).

2. Authorization based on Kerberos V5.

-k Either one of the following must succeed. The order in which the authorization checks are done is as specified below.

1. Authorization based on Kerberos V5.
2. Authentication based on privileged port numbers and authorization of the remote user through equivalent accounts.

Note: The **-k** option is ignored when used with **-K**, and the **-r** option is ignored when used with **-R**. Also, if no options are specified, the default option is **-K**.

Operation

When **remshd** receives a service request, it responds with the following protocol:

1. The server checks the client's source port. If the port is not a privileged port, i.e., in the range 512 through 1023, and **remshd** is operating in a non-secure environment, the connection is terminated. In a secure environment, the action taken depends on the command line options:

   -R The source port must be a privileged port otherwise the connection is terminated.

   -r If the source port is not a privileged port then authorization based on Kerberos must succeed or the connection is terminated.

   -k The source port must be a privileged port if Kerberos authorization fails.

   -K No action is taken.

2. The server reads characters from the connection up to a null (**\0**) byte. It interprets the resulting string as an ASCII number, base 10.

3. If the number is non-zero, it is interpreted as the port number of a secondary stream to be used for standard error. A second connection is then created to the specified port on the client's host. (The source port of this second connection will also be checked as specified in item 1.) If the first character sent is a null (**\0**), no secondary connection is made, and the standard error from the command is sent to the primary stream. If the secondary connection has been made, **remshd** interprets bytes it receives on that socket as signal numbers and passes them to the command as signals. See *signal*(2).

4. The server checks the client's source address and requests the corresponding host name (see *named*(1M), *gethostbyaddr*(3N), and *hosts*(4)). If it cannot determine the hostname, it uses the dot-notation representation of the host address.

5. In a secure environment, **remshd** performs authentication based on Kerberos V5. See *sis*(5) for details.

6. The server reads the client's host account name from the first connection. This is a null-terminated sequence not exceeding 16 characters.

7. The server reads the server's host account name from the first connection. This is a null-terminated sequence not exceeding 16 characters.

8. The server reads a command to be passed to the shell from the first connection. The command length is limited by the maximum size of the system's argument list.

9. **remshd** then validates the user as follows (all actions take place on the host **remshd** runs on):

   a. It looks up the user account name (retrieved in step 6) in the password file. If it finds it, it performs a **chdir**() to either the user's home directory, if there is one, or to "/."

   b. If either the lookup or **chdir**() fails, the connection is terminated (see *chdir*(2)).

   c. The connection is also terminated if

      - the account accessed is administratively locked. The account can be locked by entering a character in the password field that is not part of the set of digits (such as *). The characters used to represent "digits" are . for 0, / for 1, 0 through 9 for 2 through 11, A through Z for 12 through 37, and a through z for 38 through 63. (See also *passwd*(4)).

      - in a non-secure environment, the account accessed is protected by a password and, either the password expired or the account on the client's host is not equivalent to the account accessed.

r

- in a secure environment, the command line options decide whether connection is to be terminated.

  -K if Kerberos authorization does not succeed the connection is terminated (see *sis*(5) for details on authorization).

  -R if the client's host is not equivalent to the account accessed, the connection is terminated.

  -r if the account is not equivalent to the account accessed, then Kerberos authorization has to succeed or the connection is terminated.

  -k if Kerberos authorization fails, then the account has to be equivalent or the connection is terminated. For more information on equivalent accounts, see *hosts.equiv*(4).

10. A null byte is returned on the primary connection and the command line is passed to the normal login shell of the user with that shell's **-c** option. The shell inherits the network connections established by **remshd** and assumes the normal user and group permissions of the user.

    remshd uses the following path when executing the specified command:

    /usr/bin:/usr/ccs/bin:/usr/bin/X11:/usr/contrib/bin:/usr/local/bin

11. If a secondary socket has been set up, **remshd** normally exits when command standard error and secondary socket standard error have both been closed. If no secondary socket was set up, **remshd** has called an *exec*(2) function, launched the command process, and is no longer present.

DIAGNOSTICS

All diagnostic messages are returned on the connection associated with standard error after which any network connections are closed. An error is indicated by a leading byte with a value of 1 (0 is returned in step 9 above upon successful completion of all the steps before the command execution).

Malformed from address

The first socket connection does not use a reserved port or the client's host address is not an Internet address.

Can't get stderr port

Unable to complete the connection of the secondary socket used for error communication.

Second port not reserved

The secondary socket connection does not use a reserved port.

Locuser too long

The name of the user account on the client's host is longer than 16 characters.

Remuser too long

The name of the user on the server's host is longer than 16 characters.

Command too long

The command line passed exceeds the size of the argument list (as configured into the system).

Login incorrect

No password file entry existed for the user name on the server's host, or the authentication procedure described above in step 8 failed.

No remote directory

The **chdir** command to the home directory or "/" on the server's host failed.

Can't make pipe

The pipe needed for the standard error output wasn't created.

No more processes

The server was unable to fork a process to handle the incoming connection.

Next step: Wait a period of time and try again. If this message persists, the server's host may have runaway processes that are using all the entries in the process table.

system call: *message*

Error in executing the named system call. The message specifies the cause of the failure.

shellname: ...

The user's login shell could not be started. This message is returned on the connection associated with the standard error, and is not preceded by a leading byte with a value of 1. Other messages can be returned by the remote command when it executes.

rcmd: connect : <hostname>: Connection refused.

This generic message could be due to a number of reasons. One of the reasons could be because the entry for *shell* service is not present in **/etc/inetd.conf**. This entry may have been removed or commented out to prevent non-secure access.

Kerberos specific errors are listed in *sis*(5).

WARNINGS

The integrity of each host and the connecting medium is assumed if the "privileged port" authentication procedure is used in a non-secure environment or if the command line options **-R** or **-r** are used in a secure environment. Although both these methods provide insecure access, they are useful in an "open" environment.

Note also that all information, including any passwords, are passed unencrypted between the two hosts when **remshd** is invoked in a non-secure environment.

remshd ignores **SIGHUP**, **SIGINT**, **SIGQUIT**, and **SIGTERM**, so these signal numbers can safely be sent to remote commands via the secondary socket provided by **remshd**. Other signal numbers may cause **remshd** to kill itself.

AUTHOR

remshd was developed by the University of California, Berkeley.

FILES

$HOME/.rhosts	User's private equivalence list
/etc/hosts.equiv	List of equivalent hosts

SEE ALSO

remsh(1), inetd(1M), named(1M), rcmd(3N), hosts(4), hosts.equiv(4), inetd.conf(4), inetd.sec(4), services(4), sis(5).

NAME

renice - alter priority of running processes

SYNOPSIS

renice [**-n** *newoffset*] [**-g**|**-p**|**-u**] *id* ...

DESCRIPTION

The **renice** command alters the system nice value (used in the system scheduling priority) of one or more running processes specified by *id* .... The new system nice value is set to 20 + *newoffset*, and is limited to the range 0 to 39. However if the **UNIX95** environment variable is set, the new system nice value is set to current nice value + *newoffset.* Processes with lower system nice values run at higher system priorities than processes with higher system nice values. The **-l** option of the **ps** command shows the current priority (**PRI**) and nice value (**NI**) for processes. See also *nice*(1).

To reduce the system nice value of a process, or to set it to a value less than 20 (with a negative *newoffset*), a user must have appropriate privileges. Otherwise, users cannot decrease the system nice value of a process and can only increase it within the range 20 to 39, to prevent overriding any current administrative restrictions.

To alter the system nice value of another user's process, a user must have appropriate privileges. Otherwise, users can only affect processes that they own.

Options

renice recognizes the following options. If no **-g**, **-p**, or **-u** option is specified, the default is **-p**.

-g *id* ... Interpret each *id* as a process group ID. All processes in each process group have their system nice value altered. Only users with appropriate privileges can use this option.

-n *newoffset* Change the system nice value of each affected process to 20 + *newoffset*. If the **UNIX95** environment variable is set, the system nice value of each affected process is changed to current nice value + *newoffset.*

If *newoffset* is negative, the system nice value is set to 20 minus the absolute value of *newoffset*. If the **UNIX95** environment variable is set and the *newoffset* is negative, the system nice value is set to current nice value minus the absolute value of *newoffset.* Only users with appropriate privileges can reduce the system nice value or set it to less than 20. If this option is omitted, *newoffset* defaults to 10.

-p *id* ... Interpret each *id* as a process ID. This is the default.

Note: *id* is a process ID as reported by the **ps** command, not a job number (e.g., **%1**), as used by some shells.

-u *id* ... Interpret each *id* as a user name or user ID number. All processes owned by each specified user have their system nice values altered. Only users with appropriate privileges can use this option for user names and IDs other than their own.

r

RETURN VALUES

renice returns a 0 when successful, and a non-zero value when unsuccessful.

EXTERNAL INFLUENCES

Single-byte character code sets are supported.

DIAGNOSTICS

renice reports the old and new *newoffset* values (system nice value – 20) of the affected processes if the operation requested completes successfully. Otherwise, an error message is displayed to indicate the reason for failure.

However, if the **UNIX95** environment variable is set, no reporting is done unless the command fails.

EXAMPLES

Use **renice** default values to decrease the priority of process **923**. The *id* type defaults to **-p**, and *newoffset* defaults to **10**, setting the process to a system nice value of 30.

```
renice 923
```

Change the system nice value for all processes owned by user **john** and user **123** to 33 (*newoffset*=13). (Affecting other users processes requires appropriate privileges.)

```
renice -n 13 -u john 123
```

Change the system nice value of all processes in process group 20 to **10**. (Lowering the system nice value of a process group requires appropriate privileges.)

```
renice -n -10 -g 20
```

WARNINGS

Users who do not have appropriate privileges cannot reduce the system nice values of their own processes, even if they increased them in the first place.

FILES

/etc/passwd	Maps user names to user ID's

SEE ALSO

nice(1), ps(1), getpriority(2), nice(2).

STANDARDS CONFORMANCE

renice: XPG4

NAME

repquota - summarize file system quotas

SYNOPSIS

/usr/sbin/repquota [**-v**] *filesystem* ...

/usr/sbin/repquota [**-v**] **-a**

DESCRIPTION

The **repquota** command prints a summary of disk usage and quotas for each specified *filesystem*.

filesystem is either the name of the directory on which the file system is mounted or the name of the device containing the file system.

For each user, the current number of files and amount of space (in Kbytes) is printed, along with any quotas created with **edquota** (see *edquota*(1M)).

Options

repquota recognizes the following options:

- **-a** Report on all appropriate file systems in **/etc/fstab**.
- **-v** Report all quotas, even if there is no usage.

EXTERNAL INFLUENCES

Environment Variables

LC_MESSAGES determines the language in which messages are displayed.

If **LC_MESSAGES** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**.

If any internationalization variable contains an invalid setting, **repquota** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported.

AUTHOR

Disk Quotas were developed by the University of California, Berkeley, Sun Microsystems, and HP.

FILES

/etc/fstab	Static information about the file systems
/etc/mnttab	Mounted file system table
directory/**quotas**	Quota statistics static storage for the file system, where *directory* is the root of the file system as interpreted by **mount** (see *mount*(1M)).

SEE ALSO

edquota(1M), mount(1M), quota(5).

NAME

restore, rrestore - restore file system incrementally, local or across network

SYNOPSIS

/usr/sbin/restore *key* [*name* ...]

/usr/sbin/rrestore *key* [*name* ...]

DESCRIPTION

The **restore** and **rrestore** commands read tapes previously dumped by the **dump** or **rdump** command (see *dump*(1M) and *rdump*(1M)). Actions taken are controlled by the *key* argument where *key* is a string of characters containing not more than one function letter and possibly one or more function modifiers. One or more *name* arguments, if present, are file or directory names specifying the files that are to be restored. Unless the **h** modifier is specified (see below), the appearance of a directory name refers to the files and (recursively) subdirectories of that directory.

Function Portion of *key*

The function portion of the key is specified by one of the following letters:

r Read the tape and load into the current directory. **r** should be used only after careful consideration, and only to restore a complete dump tape onto a clear file system, or to restore an incremental dump tape after a full level zero restore. Thus,

```
/usr/sbin/newfs -F hfs /dev/rdsk/c0t0d0
/usr/sbin/mount /dev/dsk/c0t0d0 /mnt
cd /mnt
restore r
```

is a typical sequence to restore a complete dump. Another **restore** or **rrestore** can then be performed to restore an incremental dump on top of this. Note that **restore** and **rrestore** leave a file **restoresymtab** in the root directory of the file system to pass information between incremental restore passes. This file should be removed when the last incremental tape has been restored. A **dump** or **rdump** followed by a **newfs** and a **restore** or **rrestore** is used to change the size of a file system (see *newfs*(1M)).

R **restore** and **rrestore** request a particular tape of a multivolume set on which to restart a full restore (see **r** above). This provides a means for interrupting and restarting **restore** and **rrestore**.

x Extract the named files from the tape. If the named file matches a directory whose contents had been written onto the tape, and the **h** modifier is not specified, the directory is recursively extracted. The owner, modification time, and mode are restored (if possible). If no file argument is given, the root directory is extracted, which results in the entire contents of the tape being extracted, unless **h** has been specified.

t Names of the specified files are listed if they occur on the tape. If no file argument is given, the root directory is listed, which results in the entire content of the tape being listed, unless **h** has been specified.

s The next argument to **restore** is used as the dump file number to recover. This is useful if there is more than one dump file on a tape.

i This mode allows interactive restoration of files from a dump tape. After reading in the directory information from the tape, **restore** and **rrestore** provide a shell-like interface that allows the user to move around the directory tree selecting files to be extracted. The available commands are given below; for those commands that require an argument, the default is the current directory.

add [*arg*] The current directory or specified argument is added to the list of files to be extracted. If a directory is specified, it and all its descendents are added to the extraction list (unless the **h** key is specified on the command line). File names on the extraction list are displayed with a leading * when listed by **ls**.

cd [*arg*] Change the current working directory to the specified argument.

delete [*arg*] The current directory or specified argument is deleted from the list of files to be extracted. If a directory is specified, it and all its descendents are deleted from the extraction list (unless **h** is specified on the command line). The most expedient way to extract files from a directory is to add the directory to the extraction list, then delete unnecessary files.

extract All files named on the extraction list are extracted from the dump tape. **restore** and **rrestore** ask which volume the user wants to mount. The fastest way to extract a few files is to start with the last volume, then work toward the first volume.

help List a summary of the available commands.

ls [*arg*] List the current or specified directory. Entries that are directories are displayed with a trailing **/**. Entries marked for extraction are displayed with a leading *****. If the verbose key is set, the inode number of each entry is also listed.

pwd Print the full path name of the current working directory.

quit **restore** and **rrestore** immediately exit, even if the extraction list is not empty.

set-modes Set the owner, modes, and times of all directories that are added to the extraction list. Nothing is extracted from the tape. This setting is useful for cleaning up after a restore aborts prematurely.

verbose The sense of the **v** modifier is toggled. When set, the verbose key causes the **ls** command to list the inode numbers of all entries. It also causes **restore** and **rrestore** to print out information about each file as it is extracted.

Function Modifiers

The following function modifier characters can be used in addition to the letter that selects the function desired:

b Specify the block size of the tape in kilobytes. If the **-b** option is not specified, **restore** and **rrestore** try to determine the tape block size dynamically.

f Specify the name of the archive instead of **/dev/rmt/0m**. If the name of the file is **-**, **restore** reads from standard input. Thus, **dump** and **restore** can be used in a pipeline to dump and restore a file system with the command

```
dump 0f - /usr | (cd /mnt; restore xf -)
```

When using **rrestore**, this key should be specified, and the next argument supplied should be of the form *machine***:***device*.

h Extract the actual directory, rather than the files to which it refers. This prevents hierarchical restoration of complete subtrees from the tape, rather than the files to which it refers.

m Extract by inode numbers rather than by file name. This is useful if only a few files are being extracted and one wants to avoid regenerating the complete path name to the file.

v Type the name of each file **restore** and **rrestore** treat, preceded by its file type. Normally **restore** and **rrestore** do their work silently; the **v** modifier specifies verbose output.

y Do not ask whether to abort the operation if **restore** and **rrestore** encounters a tape error. **restore** and **rrestore** attempt to skip over the bad tape block(s) and continue.

rrestore creates a server, either **/usr/sbin/rmt** or **/etc/rmt,** on the remote machine to access the tape device.

r

DIAGNOSTICS

restore and **rrestore** complain about bad key characters.

restore and **rrestore** complain if a read error is encountered. If the **y** modifier has been specified, or the user responds **y**, **restore** and **rrestore** attempt to continue the restore.

If the dump extends over more than one tape, **restore** and **rrestore** ask the user to change tapes. If the **x** or **i** function has been specified, **restore** and **rrestore** also ask which volume the user wants to mount. The fastest way to extract a few files is to start with the last volume and work towards the first volume.

There are numerous consistency checks that can be listed by **restore** and **rrestore**. Most checks are self-explanatory or can “never happen”. Here are some common errors:

filename: **not found on tape**
: The specified file name was listed in the tape directory but not found on the tape. This is caused by tape read errors while looking for the file, and from using a dump tape created on an active file system.

expected next file *inumber*, **got** *inumber*
: A file not listed in the directory showed up. This can occur when using a dump tape created on an active file system.

Incremental tape too low
: When doing an incremental restore, a tape that was written before the previous incremental tape, or that has too low an incremental level has been loaded.

Incremental tape too high
: When doing an incremental restore, a tape that does not begin its coverage where the previous incremental tape left off, or that has too high an incremental level has been loaded.

Tape read error while restoring *filename*
Tape read error while skipping over inode *inumber*
Tape read error while trying to resynchronize
: A tape read error has occurred. If a file name is specified, the contents of the restored files are probably partially wrong. If restore is skipping an inode or is trying to resynchronize the tape, no extracted files are corrupted, although files may not be found on the tape.

Resync restore, skipped *num* **blocks**
: After a tape read error, **restore** and **rrestore** may have to resynchronize themselves. This message indicates the number of blocks skipped over.

WARNINGS

restore and **rrestore** can get confused when doing incremental restores from dump tapes that were made on active file systems.

A level zero dump (see *dump*(1M)) must be done after a full restore. Since restore runs in user code, it has no control over inode allocation; thus a full dump must be done to get a new set of directories reflecting the new inode numbering, even though the contents of the files are unchanged.

AUTHOR

restore and **rrestore** were developed by the University of California, Berkeley.

FILES

/dev/rmt/0m	Default tape drive.
/tmp/rstdr*	File containing directories on the tape.
/tmp/rstmd*	Owner, mode, and time stamps for directories.
./restoresymtab	Information passed between incremental restores.

SEE ALSO

dump(1M), mkfs(1M), mount(1M), newfs(1M), rmt(1M).

NAME

revck - check internal revision numbers of HP-UX files

SYNOPSIS

/usr/old/usr/bin/revck *ref_files*

DESCRIPTION

revck checks the internal revision numbers of lists of files against reference lists. Each *ref_file* must contain a list of absolute path names (each beginning with **/**) and *whatstrings* (revision information strings from **what** — see *what*(1)). Path names begin in column 1 of a line, and have a colon appended to them. Each path name is followed by zero or more lines of *whatstrings*, one per line, each indented by at least one tab (this is the same format in which **what** outputs its results).

For each path name, **revck** checks that the file exists, and that executing **what** on the current path name produces results identical to the *whatstrings* in the reference file. Only the first 1024 bytes of *whatstrings* are checked.

ref_files are usually the absolute path names of the *revlist* files shipped with HP-UX. Each HP-UX software product includes a file named **/system/***product/***revlist** (for example, **/system/97070A/revlist**). The *revlist* file for each product is a reference list for the ordinary files shipped with the product, plus any empty directories on which the product depends.

FILES

/system/*product***/revlist**	lists of HP-UX files and revision numbers

DIAGNOSTICS

revck is silent except for reporting missing files or mismatches.

WARNINGS

revck produces unpredictable results if a *ref_file* is not in the right format.

SEE ALSO

what(1).

r

NAME

rexd - RPC-based remote execution server

SYNOPSIS

/usr/sbin/rpc.rexd [**-l** *log_file*] [**-m** *mountdir*] [**-r**]

DESCRIPTION

rexd is the RPC server for remote command execution. A **rexd** is started by **inetd** when a remote execution request is received (see *inetd*(1M)). **rexd** exits when command execution has completed.

If the user ID (uid) in the remote execution request is assigned to a user on the server, **rexd** executes the command as that user. If no user on the server is assigned to the uid, **rexd** does not execute the command. The **-r** option and **inetd.sec** security file allow for better access control (see *inetd.sec*(4)).

For noninteractive commands, standard output and error file descriptors are connected to sockets. Interactive commands use pseudo terminals for standard input, output, and error (see *pty*(7)).

If the file system specified in the remote execution request is not already mounted on the server, **rexd** uses NFS to mount the file system for the duration of the command execution (see *nfs*(7)). **rexd** mounts file systems with the **nosuid** and **soft** options. For more details on mount options see *mount*(1M). If the server cannot mount the file system, an error message is returned to the client. By default, any mount points required by **rexd** are created below **/var/spool/rexd**. To change the default location, use the **-m** option.

Options

rexd recognizes the following options and command-line arguments:

-l *log_file*
: Log any diagnostic, warning, and error messages to *log_file*. If *log_file* exists, **rexd** appends messages to the file. If *log_file* does not exist, **rexd** creates it. Messages are not logged if the **-l** option is not specified.

  Information logged to the file includes date and time of the error, host name, process ID and name of the function generating the error, and the error message. Note that different RPC services can share a single log file because enough information is included to uniquely identify each error.

-m *mountdir*
: Create temporary mount points below directory *mountdir*. By default, **rexd** creates temporary mount points below **/var/spool/rexd**. The directory *mountdir* should have read and execute permission for all users (mode 555). Otherwise, **rexd** denies execution for users that do not have read and execute permission.

-r
: Use increased security checking. When started with the **-r** option, **rexd** denies execution access to a client unless one of the following conditions is met:

  - The name of the client host is in **/etc/hosts.equiv** file on the server.
  - The user on the server that is associated with the uid sent by the client has an entry in **$HOME/.rhosts** specifying the client name on a line or the client name followed by at least one blank and the user's name.

    For example, assume a user whose login name is **mjk** is assigned to uid 7 on **NODE1** and executes the following **on** command:

    ```
    on NODE2 pwd
    ```

    User **mjk** on **NODE2** must have one of the following entries in **$HOME/.rhosts**:

    ```
    NODE1
    NODE1 mjk
    ```

r

DIAGNOSTICS

The following is a subset of the messages that could appear in the log file if the **-l** option is used. Some of these messages are also returned to the client.

rexd: could not umount: *dir*
: **rexd** was unable to **umount()** the user's current working file system. See

WARNINGS for more details.

rexd: mountdir (*mountdir***) is not a directory**
: The path name *mountdir*, under which temporary mount points are created, is not a directory or does not exist.

rexd: *command***: Command not found**
: **rexd** could not find *command*.

rexd: *command***: Permission denied**
: **rexd** was denied permission to execute *command*.

rexd: *command***: Text file busy**
: The executable file is currently open for writing.

rexd: *command***: Can't execute**
: **rexd** was unable to execute *command*.

rexd: root execution not allowed
: **rexd** does not allow execution as user **root**.

rexd: User id *uid* **not valid**
: The uid *uid* is not assigned to a user on the server.

rexd: User id *uid* **denied access**
: **rexd** was started with the **-r** option and the remote execution request did not meet either of the conditions required by the **-r** option.

rexd: *host* **is not running a mount daemon**
: The host *host* on which the user's current working directory is located is not running **mountd**. Therefore, **rexd** is unable to mount the required file system (see *mountd*(1M)).

rexd: not in export list for *file_system*
: The host on which the client's current working directory is located does not have the server on the export list for file system *file_system* containing the client's current working directory. Therefore, **rexd** is unable to mount the required file system.

WARNINGS

The client's environment is simulated by **rexd**, but not completely recreated. The simulation of the client's environment consists of mounting the file system containing the client's current working directory (if it is not already mounted) and setting the user's environment variables on the server to be the same as the user's environment variables on the client. Therefore a command run by **rexd** does not always have the same effect as a command run locally on the client.

The **rex** protocol only identifies the client user by sending the uid of the client process and the host name of the client. Therefore, it is very difficult for **rexd** to perform user authentication. If a user on the server is assigned to the uid sent by the client, **rexd** executes the requested command as that user. If no user on the client is assigned to the uid sent by the client, **rexd** returns an error.

The **-r** option has been added to provide increased user authentication. However, the authentication provided is not foolproof, and is limited by the information passed by the **rex** protocol.

In order to simulate the client's environment, **rexd** mounts the file system containing the client's current working directory (if it is not already mounted). This mount is intended to be temporary for the duration of the command.

If **rexd** mounts a file system, it attempts to **umount()** the file system after the command has completed executing. However, if **rexd** receives a **SIGKILL** signal (see *signal*(2)), the file system is not unmounted. The file system remains mounted until the superuser executes the appropriate **umount** command or the server is rebooted.

rexd's attempt to umount the file system can also fail if the file system is busy. The file system is busy if it contains an open file or a user's current working directory. The file system remains mounted until the superuser executes the appropriate **umount** command or the server is rebooted.

For more information on **rexd** security issues, see *Using and Administering NFS Services*. Security issues and their consequences should be considered before configuring **rexd** to run on a system.

FILES

/dev/pty[**pqr**]*****	Master pseudo terminals.
/dev/tty[**pqr**]*****	Slave pseudo terminals.
/dev/ptym/pty[**pqr**]*****	Master pseudo terminals.
/dev/pty/tty[**pqr**]*****	Slave pseudo terminals.
/etc/inetd.conf	Configuration file for *inetd*(1M).
/etc/hosts.equiv	List of equivalent hosts.
$HOME/.rhosts	User's private equivalence list.
/var/spool/rexd/rexd*xxxxx*	Temporary mount points for remote file systems where *xxxxx* is a string of alpha numeric characters.

AUTHOR

rexd was developed by Sun Microsystems, Inc.

SEE ALSO

on(1), inetd(1M), mount(1M), exports(4), inetd.conf(4), inetd.sec(4).

Using and Administering NFS Services

NAME

rexecd - remote execution server

SYNOPSIS

/usr/lbin/rexecd [**-n**] [**-s**]

DESCRIPTION

rexecd is the server for the *rexec*(3N) routine; it expects to be started by the internet daemon (see *inetd*(1M)). **rexecd** provides remote execution facilities with authentication based on user account names and unencrypted passwords.

inetd(1M) calls **rexecd** when a service request is received at the port indicated for the "exec" service specification in **/etc/services**; see *services*(4). To run **rexecd**, the following line should be present in **/etc/inetd.conf**:

```
exec  stream  tcp  nowait  root  /usr/lbin/rexecd  rexecd
```

See *inetd.conf*(4) for more information.

Options:

rexecd recognizes the following options.

-n Disable transport-level keep-alive messages. By default, the messages are enabled. The keep-alive messages allow sessions to time out if the client crashes or becomes unreachable.

-s This option is used in multi-homed NIS systems. It disables **remshd** from doing a reverse lookup of the client's IP address; see *gethostbyname*(3N) for more information. It can be used to circumvent an NIS limitation with multi-homed hosts.

When a service request is received, the following protocol is initiated:

1. The server reads characters from the socket up to a null (**\0**) byte. The resultant string is interpreted as an ASCII number, base 10.
2. If the number received in step 1 is non-zero, it is interpreted as the port number of a secondary stream to be used for the **stderr**. A second connection is then created to the specified port on the client's host. If the first character sent is a null (**\0**), no secondary connection is made and the **stderr** of the command is sent to the primary stream. If the secondary connection has been made, **rexecd** interprets bytes it receives on that socket as signal numbers and passes them to the command as signals (see *signal*(2)).
3. A null-terminated user name of not more than 16 characters is retrieved on the initial socket.
4. A null-terminated, unencrypted, password of not more than 16 characters is retrieved on the initial socket.
5. A null-terminated command to be passed to a shell is retrieved on the initial socket. The length of the command is limited by the upper bound on the size of the system's argument list.
6. **rexecd** then validates the user as is done by **login** (see *login*(1)). But it does not use any PAM modules of **login** for authentication. If the authentication succeeds, **rexecd** changes to the user's home directory and establishes the user and group protections of the user. If any of these steps fail, **rexecd** returns a diagnostic message through the connection, then closes the connection.
7. A null byte is returned on the connection associated with **stderr** and the command line is passed to the normal login shell of the user with that shell's **-c** option. The shell inherits the network connections established by **rexecd**.

rexecd uses the following path when executing the specified command:

```
/usr/bin:/usr/ccs/bin:/usr/bin/X11:/usr/contrib/bin:/usr/local/bin
```

Transport-level keepalive messages are enabled unless the **-n** option is present. The use of keepalive messages allows sessions to be timed out if the client crashes or becomes unreachable.

DIAGNOSTICS

All diagnostic messages are returned on the connection associated with the **stderr**, after which any network connections are closed. An error is indicated by a leading byte with a value of 1 (0 is returned in step 7 above upon successful completion of all the steps prior to the command execution).

`Username too long`
The user name is longer than 16 characters.

`Password too long`
The password is longer than 16 characters.

`Command too long`
The command line passed exceeds the size of the argument list (as configured into the system).

`Login incorrect`
No password file entry for the user name existed or the wrong password was supplied.

`No remote directory`
The **`chdir`** command to the home directory failed.

`No more processes`
The server was unable to fork a process to handle the incoming connection.

Next step: Wait a period of time and try again. If the message persists, then the server's host may have a runaway process that is using all the entries in the process table.

shellname`: ...`
The user's login shell could not be started via *exec*(2) for the given reason.

WARNINGS

The password is sent unencrypted through the socket connection.

AUTHOR

`rexecd` was developed by the University of California, Berkeley.

SEE ALSO

remsh(1), inetd(1M), rexec(3N), inetd.conf(4), inetd.sec(4), services(4).

NAME

ripquery - query RIP gateways

SYNOPSIS

ripquery [**-1**] [**-2**] [**-**[**a5**] *authkey*] [**-n**] [**-N** *dest*[**/***mask*]] [**-p**] [**-r**] [**-v**] [**-w** *time*] *gateway* ...

DESCRIPTION

ripquery is used to request all routes known by a RIP gateway by sending a RIP request or POLL command. The routing information in any routing packets returned is displayed numerically and symbolically. **ripquery** is intended to be used as a tool for debugging gateways, not for network management. SNMP is the preferred protocol for network management.

ripquery by default uses the RIP POLL command, which is an undocumented extension to the RIP specification supported by **routed** on SunOS 3.x and later and by **gated** 1.4 and later. The RIP POLL command is preferred over the RIP REQUEST command because it is not subject to Split Horizon and/or Poisoned Reverse. See the RIP RFC for more information.

Options

-1 Send the query as a version 1 packet.

-2 Send the query as a version 2 packet (default).

-[**a5**]*authkey* Specifies the authentication password to use for queries. If **-a** specified, an authentication type of SIMPLE will be used, if **-5** is specified, an authentication type of MD5 will be used; otherwise the default is an authentication type of NONE. Authentication fields in incoming packets will be displayed, but not validated.

-n Prevents the address of the responding host from being looked up to determine the symbolic name.

-N *dest*[**/***mask*]
Specifies that the query should be for the specified *dest***/***mask* instead of complete routing table. The specification of the optional mask implies a version 2 query. Up to 23 requests about specific destinations may be include in one packet.

-p Uses the RIP POLL command to request information from the routing table. This is the default, but is an undocumented extension supported only by some versions of unOS 3.x and later versions of **gated**. If there is no response to the RIP POLL command, the RIP REQUEST command is tried. **gated** responds to a POLL command with all the routes learned via RIP.

-r Used the RIP REQUEST command to request information from the gateway's routing table. Unlike the RIP POLL command, all gateways should support the RIP REQUEST. If there is no response to the RIP REQUEST command, the RIP POLL command is tried. **gated** responds to a REQUEST command with all the routes he announces out the specified interface. Due to limitations in the UDP interface, on systems based on BSD 4.3 Reno or earlier, REQUESTs respond about the interface used to route packets back to the sender. This can be avoided by running **ripquery** on the host being queried.

-v Version information about **ripquery** is displayed before querying the gateways.

-w *time* Specifies the time in seconds to wait for the initial response from a gateway. The default value is 5 seconds.

AUTHORS

Jeffrey C Honig.

SEE ALSO

gated(1M), gdc(1M), ospf_monitor(1M), *GateD Documentation*, *GateD Configuration Guide*.

BUGS

Some versions of Unix do not allow looking up the symbolic name of a subnet.

NAME

rlogind - remote login server

SYNOPSIS

/usr/lbin/rlogind [**-lns**] [**-B** *bannerfile*]

DESCRIPTION

rlogind is the server for the *rlogin*(1) program. It provides a remote login facility with authentication based on privileged port numbers. **rlogind** expects to be executed by the Internet daemon (*inetd*(1M)) when it receives a service request at the port indicated in the services database for **login** using the **tcp** protocol (see *services*(4)).

When a service request is received, the following protocol is initiated by **rlogind**:

1. **rlogind** checks the client's source port. If the port is not in the range 512 through 1023 (a "privileged port"), the server aborts the connection.
2. **rlogind** checks the client's source address and requests the corresponding host name (see *gethostent*(3N), *hosts*(4), and *named*(1M)). If it cannot determine the hostname, it uses the Internet dot-notation representation of the host address.

Once the source port and address have been checked, **rlogind** proceeds with the authentication process described in *hosts.equiv*(4). **rlogind** then allocates a STREAMS based pseudo-terminal (see *ptm*(7), *pts*(7)), and manipulates file descriptors so that the slave half of the pseudo-terminal becomes **stdin**, **stdout**, and **stderr** for a login process. The login process is an instance of *login*(1) invoked with the **-f** option if authentication has succeeded. If automatic authentication fails, *login*(1) prompts the user with the normal login sequence.

Options

The **rlogind** command supports the following options:

-l This option is used to prevent any authentication based on the user's **.rhosts** file unless the user is logging in as super-user.

-s This option is used in multi-homed NIS systems. It disables **rlogind** from doing a reverse lookup, of the client's IP address; see *gethostbyname*(3N). It can be used to circumvent an NIS limitation with multihomed hosts.

-n This option is used to disable transport-level keepalive messages.

-B *bannerfile*
This option is used to display the file *bannerfile* to incoming **rlogin** requests.

The **rlogind** process manipulates the master side of the pseudo-terminal, operating as an intermediary between the login process and the client instance of the **rlogin** program. The protocol described in *ptm*(7) and *pts*(7) is used to enable and disable flow control via Ctrl-S/Ctrl-Q under the direction of the program running on the slave side of the pseudo-terminal, and to flush terminal output in response to interrupt signals. The login process sets the baud rate and **TERM** environment variable to correspond to the client's baud rate and terminal type (see *environ*(5)).

Transport-level keepalive messages are enabled unless the **-n** option is present. The use of keepalive messages allows sessions to be timed out if the client crashes or becomes unreachable.

To start **rlogind** from the Internet daemon, the configuration file **/etc/inetd.conf** must contain an entry as follows:

```
login   stream   tcp   nowait   root   /usr/lbin/rlogind   rlogind
```

EXTERNAL INFLUENCES

International Code Set Support

Single- and multibyte character code sets are supported.

DIAGNOSTICS

Errors in establishing a connection cause an error message to be returned with a leading byte of 1 through the socket connection, after which the network connection is closed. Any errors generated by the login process or its descendents are passed through by the server as normal communication.

fork: No more processes
The server was unable to fork a process to handle the incoming connection.

Next step: Wait a period of time and try again. If this message persists, the server's host may have runaway processes that are using all the entries in the process table.

`Cannot allocate pty on remote host`

The server was unable to obtain a pseudo-terminal for use with the login process. Either all pseudo-terminals were in use, or the pty driver has not been properly set up. Note, the number of slave devices that can be allocated depends on NSTRPTY, a kernel tunable parameter. This can be changed via SAM (see *ptm*(7), *pts*(7)).

Next step: Check the pty configuration of the host where **`rlogind`** executes.

`Permission denied`

The server denied access because the client was not using a reserved port. This should only happen to interlopers trying to break into the system.

`/usr/bin/login: ...`

The login program could not be started via *exec*(2) for the reason indicated.

Next step: Try to correct the condition causing the problem. If this message persists, contact your system administrator.

WARNINGS

The "privileged port" authentication procedure used here assumes the integrity of each host and the connecting medium. This is insecure, but is useful in an "open" environment. Note that any passwords are sent unencrypted through the socket connection.

AUTHOR

`rlogind` was developed by the University of California, Berkeley.

FILES

`/etc/hosts.equiv`	List of equivalent hosts
`$HOME/.rhosts`	User's private equivalence list

SEE ALSO

login(1), rlogin(1), inetd(1M), named(1M), gethostent(3N), ruserok(3N), hosts(4), hosts.equiv(4), inetd.conf(4), services(4), environ(5), pty(7).

NAME

rlogind - remote login server

SYNOPSIS

/usr/lbin/rlogind [**-ln**] [**-B** *bannerfile*]

Kerberos V5 Network Authentication environments:

/usr/lbin/rlogind [**-clnKkRr**] [**-B** *bannerfile*]

DESCRIPTION

rlogind is the server for the *rlogin*(1) program. It provides a remote login facility with two kinds of authentication methods:

1. Authentication based on privileged port numbers where the client's source port must be in the range 512 through 1023. In this case **rlogind** assumes it is operating in normal or non-secure environment.
2. Authentication based on Kerberos V5. In this case **rlogind** assumes it is operating in a Kerberos V5 Network Authentication, i.e., secure environment.

The *inetd* daemon invokes rlogind if a service request is received at ports indicated by the **login** or **klogin** services specified in **/etc/services** (see *inetd*(1M) and *services*(4)). Service requests arriving at the **klogin** port assume a secure environment and expect Kerberos authentication to take place.

To start **rlogind** from the *inetd* daemon in a non-secure environment, the configuration file **/etc/inetd.conf** must contain an entry as follows:

```
login  stream  tcp  nowait  root  /usr/lbin/rlogind  rlogind
```

In a secure environment, **/etc/inetd.conf** must contain an entry:

```
klogin  stream  tcp  nowait  root  /usr/lbin/rlogind  rlogind -K
```

See *inetd.conf*(4) for more information.

To prevent non-secure access, the entry for **login** should be commented out in **/etc/inetd.conf**. Any non-Kerberos access will be denied since the entry for the port indicated by **login** has now been removed or commented out. In a such a situation, a generic error message,

rcmd: connect <*hostname***> : Connection refused**

is displayed. See *DIAGNOSTICS* for more details.

Options

rlogind recognizes the following options:

-c Ignore checksum verification. This option is used to achieve interoperability between clients and servers using different checksum calculation methods. For example, the checksum calculation in a application developed with Kerberos V5 Beta 4 API is different from the calculation in a Kerberos V5-1.0 application.

-l Prevents any authentication based on the user's **.rhosts** file unless the user is logging in as super-user.

-B*bannerfile*
Causes the file, *bannerfile*, to be displayed to incoming rlogin requests.

In a secure environment, **rlogind** will recognize the following additional options:

-K Authorization based on Kerberos V5 must succeed or access will be rejected (see *sis*(5) for details on authorization).

-R Authentication based on privileged port numbers and authorization of the remote user through equivalent accounts must succeed. For more information on equivalent accounts, see *hosts.equiv*(4).

-r Either one of the following must succeed. The order in which the authorization checks are done is as specified below.

1. Authentication based on privileged port numbers and authorization of the remote user through equivalent accounts (see *hosts.equiv*(4)).

r

2. Authorization based on Kerberos V5.

-k Either one of the following must succeed. The order in which the authorization checks are done is as specified below.

1. Authorization based on Kerberos V5.
2. Authentication based on privileged port numbers and authorization of the remote user through equivalent accounts.

Note: The **-k** option is ignored when used with **-K**, and the **-r** option is ignored when used with **-R**. Also, if no options are specified, the default option is **-K**.

Operation

When a service request is received, the following protocol is initiated by **rlogind**:

1. **rlogind** checks the client's source port. If the port is not in a privileged port, i.e., in the range 512 through 1023, and **rlogind** is operating in a non-secure environment, the connection is terminated. In a secure environment, the action taken depends on the command line options:

   -R The source port must be a privileged port otherwise **rlogind** terminates the connection.

   -r If the source port is not a privileged port then Kerberos authorization must succeed or the connection is terminated.

   -k The source port must be a privileged port if Kerberos authorization fails.

   -K No action is taken.

2. **rlogind** checks the client's source address and requests the corresponding host name (see *gethostent*(3N), *hosts*(4), and *named*(1M)). If it cannot determine the hostname, it uses the Internet dot-notation representation of the host address.

3. **rlogind**, in a secure environment, proceeds with the Kerberos authentication process described in *sis*(5). If authentication succeeds, then the authorization selected by the command line option **-K**, **-R**, **-k**, or **-r** is performed. The authorization selected could be as specified in *hosts.equiv*(4) or Kerberos authorization as specified in *sis*(5).

4. **rlogind** then allocates a STREAMS based pseudo-terminal (see *ptm*(7), *pts*(7)), and manipulates file descriptors so that the slave half of the pseudo-terminal becomes **stdin**, **stdout**, and **stderr** for a login process.

5. This login process is an instance of *login*(1) invoked with the **-f** option if authentication has succeeded. In a non-secure environment, if automatic authentication fails, *login*(1) prompts the user with the normal login sequence. In a secure environment, if authentication fails, **rlogind** generates an error message and quits.

r

The **rlogind** process manipulates the master side of the pseudo-terminal, operating as an intermediary between the login process and the client instance of the **rlogin** program. The protocol described in *ptm*(7) and *pts*(7) is used to enable and disable flow control via Ctrl-S/Ctrl-Q under the direction of the program running on the slave side of the pseudo-terminal, and to flush terminal output in response to interrupt signals. The login process sets the baud rate and **TERM** environment variable to correspond to the client's baud rate and terminal type (see *environ*(5)).

Transport-level keepalive messages are enabled unless the **-n** option is present. The use of keepalive messages allows sessions to be timed out if the client crashes or becomes unreachable.

EXTERNAL INFLUENCES

International Code Set Support

Single- and multibyte character code sets are supported.

DIAGNOSTICS

Errors in establishing a connection cause an error message to be returned with a leading byte of 1 through the socket connection, after which the network connection is closed. Any errors generated by the login process or its descendents are passed through by the server as normal communication.

fork: No more processes

The server was unable to fork a process to handle the incoming connection.

Next step: Wait a period of time and try again. If this message persists, the server's host may have runaway processes that are using all the entries in the process table.

`Cannot allocate pty on remote host`

The server was unable to obtain a pseudo-terminal for use with the login process. Either all pseudo-terminals were in use, or the pty driver has not been properly set up. Note, the number of slave devices that can be allocated depends on NSTRPTY, a kernel tunable parameter. This can be changed via SAM (see *ptm*(7), *pts*(7)).

Next step: Check the pty configuration of the host where **`rlogind`** executes.

`Permission denied`

The server denied access because the client was not using a reserved port. This should only happen to interlopers trying to break into the system.

`/usr/bin/login: ...`

The login program could not be started via *exec*(2) for the reason indicated.

Next step: Try to correct the condition causing the problem. If this message persists, contact your system administrator.

`rcmd: connect : <hostname>: Connection refused.`

This generic message could be due to a number of reasons. One of the reasons could be because the entry for *login* service is not present in **`/etc/inetd.conf`**. This entry may have been removed or commented out to prevent non-secure access.

Kerberos specific errors are listed in *sis*(5).

WARNINGS

The integrity of each host and the connecting medium is assumed if the "privileged port" authentication procedure is used in a non-secure environment or if the command line options **`-R`** or **`-r`** are used in a secure environment. Although both these methods provide insecure access, they are useful in an "open" environment. This is insecure, but is useful in an “open” environment.

Note also that all information, including any passwords, are passed unencrypted between the two hosts when **`rlogind`** is invoked in a non-secure environment.

AUTHOR

`rlogind` was developed by the University of California, Berkeley.

FILES

`/etc/hosts.equiv`	List of equivalent hosts
`$HOME/.rhosts`	User’s private equivalence list

SEE ALSO

login(1), rlogin(1), inetd(1M), named(1M), gethostent(3N), ruserok(3N), hosts(4), hosts.equiv(4), inetd.conf(4), services(4), environ(5), pty(7), sis(5).

NAME

rlp - send LP line printer request to a remote system

SYNOPSIS

/usr/sbin/rlp -I*id* [**-C** *class*] [**-J** *job*] [**-T** *title*] [**-i**[*numcols*]] [-*kfont*] [**-w** *num*]
[**-cdfghlnptv**] *file*

DESCRIPTION

rlp transfers a spooling request to a remote system to be printed. **rlp** communicates with a spooling daemon on a remote system to transfer the spooling request. Options can be set only on the original system. Transfers of a remote request use only the **-I** option and the file.

This command is intended to be used only by the spool system in response to the **lp** command and should not be invoked directly (see *lp*(1)).

Options

rlp recognizes the following options and command-line arguments:

-I*id*	The argument *id* is the request ID.
-C *class*	Take the *class* argument as a job classification for use on the banner page.
-J *job*	Take the *job* argument as the job name to print on the banner page. Normally, the first file's name is used.
-T *title*	Use the *title* argument as the title used by **pr** instead of the file name (see *pr*(1)). **-T** is ignored unless the **-p** option is specified.
-h	Suppress the printing of the banner page.
-i[*numcols*]	Cause the output to be indented. If the next argument is numeric, it is used as the number of blanks to be printed before each line; otherwise, 8 characters are printed.
-*kfont*	Specify a *font* to be mounted on font position *k*, where *k* is from **1** through **4**.
-w*num*	Use the *num* argument number as the page width for **pr**.

The following single-letter options are used to notify the line printer spooler that the files are not standard text files. The spooling system uses the appropriate filters (if the option is supported) to print the data accordingly. These options are mutually exclusive.

-c	The files are assumed to contain data produced by *cifplot*.
-d	The files are assumed to contain data from *tex* (DVI format).
-f	Use a filter that interprets the first character of each line as a standard FORTRAN carriage control character.
-g	The files are assumed to contain standard plot data as produced by the **plot** routines.
-l	Use a filter that suppresses page breaks.
-n	The files are assumed to contain data from **ditroff** (device-independent **troff**).
-p	Use **pr** to format the files.
-t	The files are assumed to contain data from **troff** (cat phototypesetter commands).
-v	The files are assumed to contain a raster image for devices such as the Benson Varian.

WARNINGS

Some remote line printer models may not support all of these options. Options not supported are silently ignored.

When **rlp** is transferring a request that originated on another system, only the **-I** option and the file is used. This saves **rlp** from having to set the various options multiple times. Specifying unused options does not produce an error.

AUTHOR

rlp was developed by the University of California, Berkeley and HP.

FILES

```
/etc/passwd
/usr/sbin/rlpdaemon
/var/spool/lp/*
/var/adm/lp/*
/etc/lp/*
/usr/lib/lp/*
```

SEE ALSO

accept(1M), enable(1), lp(1), lpadmin(1M), lpsched(1M), lpstat(1), rcancel(1M), rlpdaemon(1M), rlpstat(1M).

NAME

rlpdaemon - remote spooling line printer daemon, message write daemon

SYNOPSIS

/usr/sbin/rlpdaemon [**-i**] [**-l**] [**-L** *logfile*]

DESCRIPTION

rlpdaemon is a line printer daemon (spool area handler) for remote spool requests. **rlpdaemon** is normally invoked at boot time from the **/sbin/rc** file or started by *inetd*(1M), when necessary. **rlpdaemon** runs on a system that receives requests to be printed. **rlpdaemon** transfers files to the spooling area, displays the queue, or removes jobs from the queue.

rlpdaemon is also used as a server process to write a message on the user's terminal, upon receiving a request from a remote system.

Options

-i Prevent **rlpdaemon** from remaining after a request is processed. This is required if **rlpdaemon** is started from *inetd*(1M).

-l Cause **rlpdaemon** to log error messages and valid requests received from the network to the file **/var/adm/lp/lpd.log**. This can be useful for debugging.

-L *logfile* Change the file used for writing error conditions from the file **/var/adm/lp/lpd.log** to *logfile*.

When **rlpdaemon** is started by *inetd*(1M), access control is provided via the file **/var/adm/inetd.sec** to allow or prevent a host from making requests. When **rlpdaemon** is not started by *inetd*(1M), all requests must come from one of the machines listed in the file **/etc/hosts.equiv** or **/var/spool/lp/.rhosts**. When **/var/spool/lp/.rhosts** is used for access, the user name should be **lp**.

The following entry should exist in **/etc/services** for remote spooling:

```
printer     515/tcp     spooler
```

EXAMPLES

To start **rlpdaemon** from **/sbin/rc**, invoke the command:

```
/usr/sbin/rlpdaemon
```

To start **rlpdaemon** from **inetd**, the following line should be included in the file **/etc/inetd.conf**:

```
printer stream tcp nowait root /usr/sbin/rlpdaemon rlpdaemon -i
```

r

WARNINGS

If the remote system is the same as the local system and **rlpdaemon** was not started by *inetd*(1M), the local system name *must* be included in file **/etc/hosts.equiv**.

AUTHOR

rlpdaemon was developed by the University of California, Berkeley and HP.

FILES

/etc/hosts.equiv
/etc/services
/var/spool/lp/*
/var/adm/lp/*
/etc/lp/*
/usr/lib/lp/*
/var/adm/inetd.sec

SEE ALSO

accept(1M), enable(1), lp(1), inetd(1M), lpadmin(1M), lpsched(1M), lpstat(1), rcancel(1M), rlp(1M), rlpdaemon(1M), rlpstat(1M). hosts.equiv(4), inetd.conf(4), inetd.sec(4), services(4).

HP-UX System Administrator manuals.

NAME

rlpstat - print status of LP spooler requests on a remote system

SYNOPSIS

/usr/sbin/rlpstat [**-d** *printer*] [**-u** *user*] [*id* ...]

DESCRIPTION

rlpstat reports the status of the specified jobs or all requests associated with a user. If no arguments are specified, **rlpstat** reports on any requests currently in the queue.

For each request submitted (i.e., each invocation of **lp** — see *lp*(1)) **rlpstat** reports the request ID, user's name, total size of the request, date of the request, and, if it is being transferred, the device.

This command is intended to be used only by the spool system in response to the **lpstat** command and should not be invoked directly (see *lpstat*(1M)).

Options

rlpstat recognizes the following options and command-line arguments:

-d *printer*	Specify a particular printer. Otherwise, the default line printer is used (or the value of the **LPDEST** environment variable).
-u *user*	Status is requested on all requests for the user who executed the **rlpstat** command on the specified printer (see the **-d** option).
id	Status is requested on the specified request IDs (as returned by **lp**). All the request IDs must be for the same printer.

AUTHOR

rlpstat was developed by the University of California, Berkeley, and HP.

FILES

/var/spool/lp/*
/var/adm/lp/*
/etc/lp/*
/usr/lib/lp/*

SEE ALSO

enable(1), lp(1), lpadmin(1M), lpsched(1M), lpstat(1), rcancel(1M), rlp(1M), rlpdaemon(1M).

NAME

rmsf - remove a special (device) file

SYNOPSIS

/sbin/rmsf [**-a**|**-k**] [**-D** *directory*] [**-q**|**-v**] *special_file* ...

/sbin/rmsf [**-C** *class* | **-d** *driver*] [**-D** *directory*] **-H** *hw_path* [**-k**] [**-q**|**-v**]

DESCRIPTION

The **rmsf** command removes one or more special files from the **/dev** directory and potentially removes information about the associated device or devices with H/W type "DEVICE" (see *ioscan*(1M)) from the system.

If no options are specified, **rmsf** removes only the *special_files* specified on the command line. The **-k** option causes **rmsf** to remove the definition of the device from the system without removing any special files. The **-a** option causes **rmsf** to remove the device definition, and all special files that map to it from the **/dev** directory (or the directory specified with the **-D** option). By default, **rmsf** only removes *special_file* as given on the command line, however, when the **-a** option is used and *special_file* is an absolute path name *special_file* will be removed even if it does not reside in the **/dev** directory (or the directory specified with the **-D** option). Note that if *special_file* belongs to a node for which H/W type is not "DEVICE", the device definition will not be removed from the system and the *special_file* will be removed if it is a leaf node.

If a **-H** *hw_path* is specified, special files are removed as follows:

- If *hw_path* belongs to a node with H/W type "DEVICE," all special files mapping to devices at that hardware path and the system definition of those devices are removed.
- If *hw_path* belongs to a node for which H/W type is not "DEVICE," then, a special file is removed as follows:
  - If it is a leaf node, only special files for that node will be removed.
  - If the node has children, then a warning message will be issued and system definition of all the children devices and their special files are removed.

The **-C** and **-d** options remove only those special files that are associated with the given device driver or that belong to the given device class, respectively. This is useful when there is more than one type of special file mapped to a single hardware path.

If the **-k** option is specified, the definition of all devices at that hardware path are removed from the system, again without removing any special files.

Normally, **rmsf** displays a message as the special files are deleted for each driver. The **-q** (quiet) option suppresses the deletion message. The **-v** (verbose) option displays the deletion message and the name of each special file as it is deleted.

Note that most drivers do not support the ability to be removed from the system.

If the device being removed from the system uses a dynamically assigned major number, that number will be freed up for future allocation.

Options

rmsf recognizes the following options:

-a	Remove the definition of the device from the system along with all special files that refer to the device. This option cannot be used with **-k**.
-C *class*	Match devices that belong to a given device class, *class*. Device classes can be listed with the **lsdev** command (see *lsdev*(1M)). They are defined in files in the directory **/usr/conf/master.d**. This option cannot be used with **-d**.
-d *driver*	Match devices that are controlled by the specified device driver, *driver*. Device drivers can be listed with the **lsdev** command (see *lsdev*(1M)). They are defined in files in the directory **/usr/conf/master.d**. This option cannot be used with **-C**.
-D *directory*	Override the default device installation directory **/dev** and remove the special files from *directory* instead. *directory* must exist; otherwise, **rmsf** displays an error message and exits. See WARNINGS.

-H *hw_path* Match devices at a given hardware path, *hw-path*. Hardware paths can be listed with the **ioscan** command (see *ioscan*(1M)). A hardware path specifies the addresses of the hardware components leading to a device. It consists of a string of numbers separated by periods (**.**), such as **52** (a card), **52.3** (a target address), and **52.3.0** (a device). If a hardware component is a bus converter, the following period, if any, is replaced by a slash (**/**) as in **2**, **2/3**, and **2/3.0**.

-k Remove the definition of the device from the system, but not any special files. This option cannot be used with **-a**.

-q Quiet option. Normally, **rmsf** displays a message as each driver is removed. This option suppresses the driver message, but not error messages. See the **-v** option.

-v Verbose option. In addition to the normal processing message, display the name of each special file as it is removed. See the **-q** option. Print the names of the files as **rmsf** is removing them.

RETURN VALUE

rmsf exits with one of the following values:

0 Successful completion, including warning diagnostics.
1 Failure. An error occurred.

DIAGNOSTICS

Most of the diagnostic messages from **rmsf** are self-explanatory. Listed below are some messages deserving further clarification. Errors cause **rmsf** to halt immediately. Warnings allow the program to continue.

Errors

No such device in the system

No device in the system matched the options specified. Use **ioscan** to list the devices in the system (see *ioscan*(1M)).

special_file **is not a special file**

The file is not associated with an I/O device.

Warnings

**WARNING: The specified hardware path is BUS_NEXUS/INTERFACE type.
This will remove all the devices connected to it.**

The H/W type of the node specified by *hw_path* is BUS_NEXUS/INTERFACE. All the devices under this path will be removed.

Cannot remove *driver* **at** *hw_path*

The definition of the device located at *hw_path* and controlled by *driver* cannot be removed from the kernel. That is *driver* does not support the **unbind** function.

No device associated with *special_file*

The special file does not map to a device in the system; the file is removed unless the **-k** option was specified.

EXAMPLES

Remove the special file **mux0** from the current directory:

```
rmsf ./mux0
```

Remove the system definition of the device associated with **/dev/lp0** along with all special files that refer to the device:

```
rmsf -a /dev/lp0
```

Remove the system definitions for all devices associated with hardware path 52.6.0:

```
rmsf -k -H 52.6.0
```

WARNINGS

Most commands and subsystems assume their device files are in **/dev**, therefore the use of the **-D** option is discouraged.

Most device drivers do not support the *unbind* operation necessary to remove the device from the system.

AUTHOR

rmsf was developed by HP.

FILES

/dev/config
/etc/ioconfig
/usr/conf/master.d/*

SEE ALSO

rm(1), insf(1M), ioscan(1M), lsdev(1M), lssf(1M), mksf(1M), ioconfig(4).

NAME

rmt - remote magnetic-tape protocol module

SYNOPSIS

/usr/sbin/rmt

DESCRIPTION

rmt is a program used by the remote dump and restore programs for manipulating a magnetic tape drive through an interprocess communication (IPC) connection. The **fbackup** and **frecover** commands also use **rmt** to achieve remote backup capability (see *fbackup*(1M) and *frecover*(1M)). **rmt** is normally started up with an **rexec()** or **rcmd()** call (see *rexec*(3N) and *rcmd*(3N)).

rmt accepts requests specific to the manipulation of magnetic tapes, performs the commands, then responds with a status indication. DDS devices that emulate magnetic tapes are also supported. All responses are in ASCII and in one of two forms. Successful commands have responses of

A*number***\n**

where *number* is an ASCII representation of a decimal number. Unsuccessful commands are responded to with

E*error-number***\n***error-message***\n**

where *error-number* is one of the possible error numbers described in *errno*(2) and *error-message* is the corresponding error string as printed from a call to **perror()** (see *perror*(3C)). The protocol is comprised of the following commands:

O*device***\n***mode***\n** Open the specified *device* using the indicated *mode*. *device* is a full pathname and *mode* is an ASCII representation of a decimal number suitable for passing to **open()** (see *open*(2)). If a device is already open, it is closed before a new open is performed.

o*device***\n***mode***\n** Open the specified *device* using the indicated *mode*. *device* is a full pathname and *mode* is an ASCII representation of an octal number suitable for passing to **open()**. If a device is already open, it is closed before a new open is performed.

C*device***\n** Close the currently open device. The *device* specified is ignored.

L*whence***\n***offset***\n** Perform an **lseek()** operation using the specified parameters (see *lseek*(2)). The response value is that returned from by **lseek()**.

W*count***\n** Write data onto the open device. **rmt** reads *count* bytes from the connection, aborting if a premature end-of-file is encountered. The response value is that returned from by **write()** (see *write*(2)).

R*count***\n** Read *count* bytes of data from the open device. If *count* exceeds the size of the data buffer (10 Kbytes), it is truncated to the data buffer size. **rmt** then performs the requested **read()** and responds with **A***count-read***\n** if the read was successful. Otherwise an error is returned in the standard format. If the read was successful, the data read is then sent.

I*operation***\n***count***\n**
Perform a **MTIOCOP ioctl()** command using the specified parameters. Parameters are interpreted as ASCII representations of the decimal values to be placed in the **mt_op** and **mt_count** fields of the structure used in the **ioctl()** call. The return value is the *count* parameter when the operation is successful.

S Return the status of the open device, as obtained with a **MTIOCGET ioctl()** call. If the operation was successful, an ACK is sent with the size of the status buffer, then the status buffer is sent (in binary).

s Return the status of the open device, as obtained with a **fstat()** call. If the operation was successful, an ACK is sent with the size of the status buffer, then the status buffer is sent (in binary). **f** Return the status of the open device, as obtained with a **fstat()** call. If the operation was successful, an ACK is sent with the size of the status buffer, then the status buffer is sent in the following ASCII format:

r

> *machine*<blank>*value*<newline>
> *stat_struct_member_name*<blank>*value*<newline>

The end of the data is indicated by an ASCII NULL character. See **/usr/include/sys/stat.h** for the **struct stat** definition. In addition to the struct stat information, there is an entry in the buffer describing the machine type as returned from a **uname()** call (see *uname*(2)). In the above format "machine" is a key word. All fields except **st_spare4** of the **struct stat** are returned.

m Return the status of the open device, as obtained with a **MTIOCGET ioctl()** call. If the operation was successful, an ack is sent with the size of the status buffer, then the status buffer is sent in the following ASCII format:

> *machine*<blank>*value*<newline>
> *mtget_struct_member_name*<blank>*value*<newline>

The end of the data is indicated by an ASCII NULL character. See **/usr/include/sys/mtio.h** for the **struct mtget** definition. In addition to the struct mtget information there is an entry in the buffer describing the machine type as returned from a **uname()** call. In the above format "machine" is a keyword.

Any other command causes **rmt** to exit.

RETURN VALUE

Device status is returned in the field **mt_gstat**. **/usr/include/sys/mtio.h** contains defined macros for checking the status bits.

DIAGNOSTICS

All responses are of the form described above.

AUTHOR

rmt was developed by the University of California, Berkeley.

SEE ALSO

ftio(1), fbackup(1M), frecover(1M), dump(1M), restore(1M), rcmd(3N), rexec(3N).

WARNINGS

Use of this command for remote file access protocol is discouraged.

NAME

route - manually manipulate the routing tables

SYNOPSIS

/usr/sbin/route [**-f**] [**-n**] [**-p** *pmtu*] **add** [**net**|**host**] *destination* [**netmask** *mask*] *gateway* [*count*]

/usr/sbin/route [**-f**] [**-n**] **delete** [**net**|**host**] *destination* [**netmask** *mask*] *gateway* [*count*]

/usr/sbin/route -f [**-n**]

DESCRIPTION

The **route** command manipulates the network routing tables manually. You must have appropriate privileges.

Subcommands

The following subcommands are supported.

add	Add the specified host or network route to the network routing table. If the route already exists, a message is printed and nothing changes.
delete	Delete the specified host or network route from the network routing table.

Options and Arguments

route recognizes the following options and arguments.

-f Delete all route table entries that specify a remote host for a gateway. If this is used with one of the subcommands, the entries are deleted before the subcommand is processed.

-n Print any host and network addresses in Internet dot notation, except for the default network address, which is printed as **default**.

-p *pmtu* Specifies a path maximum transmission unit (MTU) value for a static route. The minimum value allowed is 68 bytes; the maximum is the MTU of the outgoing interface for this route. This option can be applied to both host and network routes.

net or **host** The type of *destination* address. If this argument is omitted, routes to a particular host are distinguished from those to a network by interpreting the Internet address associated with *destination*. If the *destination* has a local address part of **INADDR_ANY(0)**, the route is assumed to be to a network; otherwise, it is treated as a route to a host.

destination The destination host system where the packets will be routed. *destination* can be one of the following:

- A host name (the official name or an alias, see *gethostent*(3N)).
- A network name (the official name or an alias, see *getnetent*(3N)).
- An Internet address in dot notation (see *inet*(3N)).
- The keyword **default**, which signifies the wildcard gateway route (see *routing*(7)).

netmask *mask* The mask that will be bit-wise ANDed with *destination* to yield a net address where the packets will be routed. *mask* can be specified as a single hexadecimal number with a leading **0x**, with a dot-notation Internet address, or with a pseudo-network name listed in the network table (see *networks*(4)). The length of the *mask*, which is the number of contiguous 1's starting from the leftmost bit position of the 32-bit field, can be shorter than the default network mask for the *destination* address. (see *routing*(7)). If the **netmask** option is not given, *mask* for the route will be derived from the *netmasks* associated with the local interfaces. (see *ifconfig*(1M)). *mask* will be defaulted to the longest *netmask* of those local interfaces that have the same network address. If there is not any local interface that has the same network address, then *mask* will be defaulted to the default network mask of *destination*.

gateway The gateway through which the destination is reached. *gateway* can be one of the following:

- A host name (the official name or an alias, see *gethostent*(3N)).
- An Internet address in dot notation (see *inet*(3N)).

count An integer that indicates whether the gateway is a remote host or the local host. If the route leads to a destination through a remote gateway, *count* should be a number greater than 0. If the route leads to *destination* and the gateway is the local host, *count* should be 0. The default for *count* is zero. The result is not defined if *count* is negative.

Operation

All symbolic names specified for a *destination* or *gateway* are looked up first as a host name using **gethostbyname()**; if the host name is not found, the *destination* is searched for as a network name using **getnetbyname()**. *destination* and *gateway* can be in dot notation (see *inet*(3N)).

If the **-n** option is not specified, any host and network addresses are displayed symbolically according to the name returned by **gethostbyaddr()** and **getnetbyaddr()**, respectively, except for the default network address (printed as **default**) and addresses that have unknown names. Addresses with unknown names are printed in Internet dot notation (see *inet*(3N)).

If the **-n** option is specified, any host and network addresses are printed in Internet dot notation except for the default network address which is printed as **default**.

If the **-f** option is specified, **route** deletes all route table entries that specify a remote host for a gateway. If it is used with one of the subcommands described above, the entries are deleted before the subcommand is processed.

Path MTU Discovery is a technique for discovering the maximum size of an IP datagram that can be sent on an internet path without causing datagram fragmentation in the intermediate routers. In essence, a source host that utilizes this technique initially sends out datagrams up to the the size of the outgoing interface. The Don't Fragment (DF) bit in the IP datagram header is set. As an intermediate router that supports Path MTU Discovery receives a datagram that is too large to be forwarded in one piece to the next-hop router and the DF bit is set, the router will discard the datagram and send an ICMP Destination Unreachable message with a code meaning "fragmentation needed and DF set". The ICMP message will also contain the MTU of the next-hop router. When the source host receives the ICMP message, it reduces the path MTU of the route to the MTU in the ICMP message. With this technique, the host route in the source host for this path will contain the proper MTU.

The **-p** *pmtu* option is useful only if you know the network environment well enough to enter an appropriate *pmtu* for a host or network route. IP will fragment a datagram to the *pmtu* specified for the route on the local host before sending the datagram out to the remote. It will avoid fragmentation by routers along the path, if the *pmtu* specified in the **route** command is correct.

ping can be used to find the *pmtu* information for the route to a remote host. The *pmtu* information in the routing table can be displayed with the **netstat -r** command (see *netstat*(1)).

The loopback interface **(lo0)** is automatically configured when the system boots with the TCP/IP software. The default IP address and netmask of the loopback interface are 127.0.0.1 and 255.0.0.0, respectively.

The 127.0.0.0 loopback route is set up automatically when **lo0** is configured so that packets for any 127.*.*.* address will loop back to the local host. Users cannot add or delete any 127.*.*.* loopback routes.

Output

add *destination*: **gateway** *gateway*

The specified route is being added to the tables.

delete *destination*: **gateway** *gateway*

The specified route is being deleted from the tables.

Flags

The values of the *count* and *destination* type fields in the **route** command determine the presence of the **G** and **H** flags in the **netstat -r** display and thus the route type, as shown in the following table.

Count	Destination Type	Flags	Route Type
=0	network	**U**	Route to a network directly from the local host
>0	network	**UG**	Route to a network through a remote host gateway
=0	host	**UH**	Route to a remote host directly from the local host
>0	host	**UGH**	Route to a remote host through a remote host gateway
=0	**default**	**U**	Wildcard route directly from the local host
>0	**default**	**UG**	Wildcard route through a remote host gateway

DIAGNOSTICS

The following error diagnostics can be displayed:

add a route that already exists

The specified entry is already in the routing table.

delete a route that does not exist

The specified route was not in the routing table.

cannot update loopback route

Routes for any 127.*.*.* loopback destination cannot be added or deleted.

WARNINGS

Reciprocal **route** commands must be executed on the local host, the destination host, and all intermediate hosts if routing is to succeed in the cases of virtual circuit connections or bidirectional datagram transfers.

The HP-UX implementation of **route** does not presently support a **change** subcommand.

AUTHOR

route was developed by the University of California, Berkeley.

FILES

/etc/networks
/etc/hosts

SEE ALSO

netstat(1), ifconfig(1M), ping(1M), ndd(1M), getsockopt(2), recv(2), send(2), gethostent(3N), getnetent(3N), inet(3N), routing(7).

NAME

rpc.nisd, rpc.nisd_resolv, nisd, nisd_resolv - NIS+ service daemon

SYNOPSIS

/usr/sbin/rpc.nisd [**-ACDFhlv**] [**-Y** [**-B** [**-t** *netid*]]] [**-d** *dictionary*] [**-L** *load*] [**-S** *level*]

rpc.nisd_resolv

DESCRIPTION

The **rpc.nisd** daemon is an RPC service that implements the NIS+ service. This daemon must be running on all machines that serve a portion of the NIS+ namespace.

rpc.nisd is usually started from a system startup script.

rpc.nisd_resolv is an auxillary process that is started by **rpc.nisd** when it is invoked with **-B** option. Note that **rpc.nisd_resolv** should not be started independently.

Options

-A Authentication verbose mode. The daemon logs all the authentication related activities to *syslogd*(1M) with **LOG_INFO** priority.

-B Provide ypserv compatible DNS forwarding for NIS host requests. The DNS resolving process, **rpc.nisd_resolv**, is started and controlled by **rpc.nisd**. This option requires that the **/etc/resolv.conf** file be set up for communication with a DNS nameserver. The **nslookup** utility can be used to verify communication with a DNS nameserver. See *resolver*(4) and *nslookup*(1).

-C Open diagnostic channel on **/dev/console**.

-D Debug mode (don't fork).

-F Force the server to do a checkpoint of the database when it starts up. Forced checkpoints may be required when the server is low on disk space. This option removes updates from the transaction log that have propagated to all of the replicas.

-L *number*
Specify the "load" the NIS+ service is allowed to place on the server. The load is specified in terms of the *number* of child processes that the server may spawn. This *number must* be at least 1 for the callback functions to work correctly. The default is 128.

-S *level* Set the authorization security level of the service. The argument is a number between 0 and 2. By default, the daemon runs at security level 2.

0 Security level 0 is designed to be used for testing and initial setup of the NIS+ namespace. When running at level 0, the daemon does not enforce any access controls. Any client is allowed to perform any operation, including updates and deletions.

1 At security level 1, the daemon accepts both **AUTH_SYS** and **AUTH_DES** credentials for authenticating clients and authorizing them to perform NIS+ operations. This is not a secure mode of operation since **AUTH_SYS** credentials are easily forged. It should not be used on networks in which any untrusted users may potentially have access.

2 At security level 2, the daemon accepts only **AUTH_DES** credentials for authentication and authorization. This is the highest level of security currently provided by the NIS+ service. This is the default security level if the **-S** option is not used.

-Y Put the server into NIS (YP) compatibility mode. When operating in this mode, the NIS+ server will respond to NIS Version 2 requests using the version 2 protocol. Because the YP protocol is not authenticated, only those items that have read access to nobody (the unauthenticated request) will be visible through the V2 protocol. It supports only the standard Version 2 maps in this mode (see **-B** option and **NOTES** in *ypfiles*(4)).

-d *dictionary*
Specify an alternate dictionary for the NIS+ database. The primary use of this option is for testing. Note that the string is not interpreted, rather it is simply passed to the **db_initialize** function. See *nis_db*(3N).

-h Print list of options.

r

-t *netid* Use *netid* as the transport for communication between **rpc.nisd** and **rpc.nisd_resolv**. The default transport is **tcp**.

-v Verbose. With this option, the daemon sends a running narration of what it is doing to the syslog daemon (see *syslogd*(1M)) at **LOG_INFO** priority. This option is most useful for debugging problems with the service (see also **-A** option).

EXAMPLES

The following example sets up the NIS+ service.

```
rpc.nisd
```

The following example sets up the NIS+ service, emulating YP with DNS forwarding.

```
rpc.nisd -YB
```

EXTERNAL INFLUENCES

Environment Variables

NETPATH The transports that the NIS+ service will use can be limited by setting this environment variable (see *netconfig*(4)).

FILES

/var/nis/parent.object
This file contains an XDR encoded NIS+ object that describes the namespace above a root server. This parent namespace may be another NIS+ namespace or a foreign namespace such as one served by the Domain Name Service. It is only present on servers that are serving the root of the namespace.

/var/nis/root.object
This file contains an XDR encoded NIS+ object that describes the root of the namespace. It is only present on servers that are serving the root of the namespace.

/etc/rc.config.d/namesvrs
initialization script for NIS+

AUTHOR

rpc.nisd and **rpc.nisd_resolv** were developed by Sun Microsystems, Inc.

SEE ALSO

nis_cachemgr(1M), nisinit(1M), nissetup(1M), nslookup(1), syslogd(1M), nis_db(3N), netconfig(4), nisfiles(4), resolver(4), ypfiles(4).

NAME

rpc.nispasswdd, nispasswdd - NIS+ password update daemon

SYNOPSIS

/usr/sbin/rpc.nispasswdd [**-a** *attempts*] [**-c** *minutes*] [**-D**] [**-g**] [**-v**]

DESCRIPTION

rpc.nispasswdd daemon is an **ONC+ RPC** service that services password update requests from *nispasswd*(1) and *yppasswd*(1). It updates password entries in the **NIS+ passwd** table.

rpc.nispasswdd is normally started from a system startup script after the NIS+ server (*rpc.nisd*(1M)) has been started. **rpc.nispasswdd** will determine whether it is running on a machine that is a master server for one or more NIS+ directories. If it discovers that the host is not a master server, then it will promptly exit. It will also determine if *rpc.nisd*(1M) is running in NIS(YP) compatibility mode (the **-Y** option) and will register as **yppasswdd** for NIS(YP) clients as well.

rpc.nispasswdd will send to syslog all failed password update attempts, which will allow an administrator to determine whether someone was trying to "crack" the passwords.

rpc.nispasswdd has to be run by a superuser.

Options

-a *attempts*	Set the maximum number of attempts allowed to authenticate the caller within a password update request session. Failed attempts are processed by *syslogd*(1M) and the request is cached by the daemon. After the maximum number of allowed attempts the daemon severs the connection to the client. The default value is set to **3**.
-c *minutes*	Set the number of minutes a failed password update request should be cached by the daemon. This is the time during which if the daemon receives further password update requests for the same user and authentication of the caller fails, then the daemon will simply not respond. The default value is set to **30** minutes.
-D	Debug. Run in debugging mode.
-g	Generate **DES** credential. By default the DES credential is not generated for the user if they do not have one. By specifying this option, if the user does not have a credential, then one will be generated for them and stored in the NIS+ cred table.
-v	Verbose. With this option, the daemon sends a running narration of what it is doing to the syslog daemon. This option is useful for debugging problems.

r

RETURN VALUE

0	Success.
1	An error has occurred.

FILES

/etc/rc.config.d/namesvrs Initialization script for NIS+.

SEE ALSO

nispasswd(1), passwd(1), yppasswd(1), rpc.nisd(1M), syslogd(1M), nsswitch.conf(4).

NAME

rpcbind - universal addresses to RPC program number mapper

SYNOPSIS

rpcbind [**-d**] [**-w**]

DESCRIPTION

rpcbind is a server that converts RPC program numbers into universal addresses. It must be running on the host to be able to make RPC calls on a server on that machine.

When an RPC service is started, it tells **rpcbind** the address at which it is listening, and the RPC program numbers it is prepared to serve. When a client wishes to make an RPC call to a given program number, it first contacts **rpcbind** on the server machine to determine the address where RPC requests should be sent.

rpcbind should be started before any other RPC service. Normally, standard RPC servers are started by port monitors, so **rpcbind** must be started before port monitors are invoked.

When **rpcbind** is started, it checks that certain name-to-address translation calls function correctly. If they fail, the network configuration databases may be corrupt. Since RPC services cannot function correctly in this situation, **rpcbind** reports the condition and terminates.

rpcbind can only be started by the super-user.

Options

rpcbind recognizes the following options:

-d Run in debug mode. In this mode, **rpcbind** will not fork when it starts, will print additional information during operation, and will abort on certain errors. With this option, the name-to-address translation consistency checks are shown in detail.

-w Do a warm start. If **rpcbind** aborts or terminates on **SIGINT** or **SIGTERM**, it will write the current list of registered services to **/tmp/portmap.file** and **/tmp/rpcbind.file**. Starting **rpcbind** with the **-w** option instructs it to look for these files and start operation with the registrations found in them. This allows **rpcbind** to resume operation without requiring all RPC services to be restarted.

WARNINGS

Terminating **rpcbind** with **SIGKILL** will prevent the warm-start files from being written.

All RPC servers must be restarted if the following occurs: **rpcbind** crashes (or is killed with **SIGKILL**) and is unable to to write the warm-start files; **rpcbind** is started without the **-w** option after a graceful termination; or, the warm-start files are not found by **rpcbind**.

AUTHOR

rpcbind was developed by Sun Microsystems, Inc.

FILES

/tmp/portmap.file

/tmp/rpcbind.file

SEE ALSO

rpcinfo(1M), rpcbind(3N).

NAME

rpcinfo - report RPC information

SYNOPSIS

rpcinfo [**-m**] [**-s**] [*host*]

rpcinfo -p [*host*]

rpcinfo -T *transport host prognum* [*versnum*]

rpcinfo -l [**-T** *transport*] *host prognum* [*versnum*]

rpcinfo [**-n** *portnum*] **-u** *host prognum* [*versnum*]

rpcinfo [**-n** *portnum*] **-t** *host prognum* [*versnum*]

rpcinfo -a *serv_address* **-T** *transport prognum* [*versnum*]

rpcinfo -b [**-T** *transport*] *prognum versnum*

rpcinfo -d [**-T** *transport*] *prognum versnum*

DESCRIPTION

rpcinfo makes an RPC call to an RPC server and reports what it finds.

In the first synopsis, **rpcinfo** lists all the registered RPC services with **rpcbind** on *host*. If *host* is not specified, the local host is the default. If **-s** is used, the information is displayed in a concise format.

In the second synopsis, **rpcinfo** lists all the RPC services registered with **rpcbind**, version 2. Also note that the format of the information is different in the first and the second synopsis. This is because the second synopsis is an older protocol used to collect the information displayed (version 2 of the **rpcbind** protocol).

The third synopsis makes an RPC call to procedure **0** of *prognum* and *versnum* on the specified *host* and reports whether a response was received. *transport* is the transport which has to be used for contacting the given service. The remote address of the service is obtained by making a call to the remote **rpcbind**.

The *prognum* argument is a number that represents an RPC program number (see *rpc*(4)).

If a *versnum* is specified, **rpcinfo** attempts to call that version of the specified *prognum*. Otherwise, **rpcinfo** attempts to find all the registered version numbers for the specified *prognum* by calling version 0, which is presumed not to exist; if it does exist, **rpcinfo** attempts to obtain this information by calling an extremely high version number instead, and attempts to call each registered version. Note that the version number is required for **-b** and **-d** options.

The other ways of using **rpcinfo** are described in the *EXAMPLES* section.

r

Options

-T *transport* Specify the transport on which the service is required. If this option is not specified, **rpcinfo** uses the transport specified in the **NETPATH** environment variable, or if that is unset or null, the transport in the *netconfig*(4) database is used. This is a generic option, and can be used in conjunction with other options as shown in the *SYNOPSIS*.

-a *serv_address*

Use *serv_address* as the (universal) address for the service on *transport* to ping procedure **0** of the specified *prognum* and report whether a response was received. The **-T** option is required with the **-a** option.

If *versnum* is not specified, **rpcinfo** tries to ping all available version numbers for that program number. This option avoids calls to remote **rpcbind** to find the address of the service. The *serv_address* is specified in universal address format of the given transport.

-b Make an RPC broadcast to procedure **0** of the specified *prognum* and *versnum* and report all hosts that respond. If *transport* is specified, it broadcasts its request only on the specified transport. If broadcasting is not supported by any transport, an error message is printed. Use of broadcasting should be limited because of the potential for adverse effect on other systems.

-d	Delete registration for the RPC service of the specified *prognum* and *versnum*. If *transport* is specified, unregister the service on only that transport, otherwise unregister the service on all the transports on which it was registered. Only the owner of a service can delete a registration, except the super-user who can delete any service.
-l	Display a list of entries with a given *prognum* and *versnum* on the specified *host*. Entries are returned for all transports in the same protocol family as that used to contact the remote **rpcbind**.
-m	Display a table of statistics of **rpcbind** operations on the given *host*. The table shows statistics for each version of **rpcbind** (versions 2, 3 and 4), giving the number of times each procedure was requested and successfully serviced, the number and type of remote call requests that were made, and information about RPC address lookups that were handled. This is useful for monitoring RPC activities on *host*.
-n *portnum*	Use *portnum* as the port number for the **-t** and **-u** options instead of the port number given by **rpcbind**. Use of this option avoids a call to the remote **rpcbind** to find out the address of the service. This option is made obsolete by the **-a** option.
-p	Probe **rpcbind** on *host* using version 2 of the **rpcbind** protocol, and display a list of all registered RPC programs. If *host* is not specified, it defaults to the local host. Note that version 2 of the **rpcbind** protocol was previously known as the portmapper protocol.
-s	Display a concise list of all registered RPC programs on *host*. If *host* is not specified, it defaults to the local host.
-t	Make an RPC call to procedure 0 of *prognum* on the specified *host* using TCP, and report whether a response was received. This option is made obsolete by the **-T** option as shown in the third synopsis.
-u	Make an RPC call to procedure 0 of *prognum* on the specified *host* using UDP, and report whether a response was received. This option is made obsolete by the **-T** option as shown in the third synopsis.

EXAMPLES

To show all of the RPC services registered on the local machine use:

```
example% rpcinfo
```

To show all of the RPC services registered with **rpcbind** on the machine named **klaxon** use:

```
example% rpcinfo klaxon
```

To show whether the RPC service with program number *prognum* and version *versnum* is registered on the machine named **klaxon** for the transport TCP use:

```
example% rpcinfo -T tcp klaxon prognum versnum
```

To show all RPC services registered with version 2 of the **rpcbind** protocol on the local machine use:

```
example% rpcinfo -p
```

To delete the registration for version 1 of the **walld** (program number **100008**) service for all transports use:

```
example# rpcinfo -d 100008 1
```

or

```
example# rpcinfo -d walld 1
```

AUTHOR

rpcinfo was developed by Sun Microsystems, Inc.

SEE ALSO

rpcbind(1M), rpc(3N), netconfig(4), rpc(4).

NAME

rpr - repair parity information in an HP SCSI disk array LUN

SYNOPSIS

rpr -b *block device_file*

DESCRIPTION

rpr repairs the data parity information on a LUN in an HP SCSI disk array when the LUN is configured in a data redundant RAID-level (RAID_1, RAID_3, or RAID_5). *block* is the logical address of the data block corresponding to the parity block needing repair. *block* is specified using the **-b** parameter. *device_file* is the name of the device file for the LUN.

Use **scn** (see *scn*(1M)) to identify data blocks that do not have correct parity blocks.

RETURN VALUE

rpr returns the following values:

0	Successful completion.
-1	Command failed (an error occurred).

DIAGNOSTICS AND ERRORS

Errors can originate from problems with:

- **rpr**
- SCSI (device level) communications
- system calls

Error messages generated by rpr:

usage: rpr -b block> <special>
rpr encountered an error in command syntax. Re-enter the command with all required arguments, in the order shown.

rpr: LUN does not exist
The addressed LUN is not configured, and is not known to the array controller.

rpr: LUN # too big
The LUN number, derived from the device file name, is out of range.

rpr: Not a raw file
rpr must be able to open the device file for raw access.

rpr: Transfer length error
The amount of data actually sent to or received from the device was not the expected amount.

rpr: Not an HP SCSI disk array
The device being addressed is not an HP SCSI disk array.

SCSI (device level) communication errors:

Sense data associated with the failed operation is printed.

Error messages generated by system calls:

rpr uses the following system calls:

malloc(), **free()**, **stat()**, **open()**, **close()**, **fopen()**, **fclose()**, **read()**, **write()**, **unlink()**, and **ioctl()**.

Documentation for these HP-UX system calls contains information about the specific error conditions associated with each call. **rpr** does not alter the value of **errno**. The interpretation of **errno** for printing purposes is performed by the system utility **strerror()**.

EXAMPLES

To repair block 12345 of the LUN **/dev/rdsk/c2t6d0** on a series 800:

```
rpr -b 12345 /dev/rdsk/c2t6d0
```

DEPENDENCIES

The HP C2425 and HP C2427 disk arrays are only supported on Series 700 systems running HP-UX version 9.0X.

The HP C2430 disk array is supported on Series 700 and 800 systems running HP-UX versions 9.0X and 10.0X.

AUTHOR

rpr was developed by HP.

SEE ALSO

scn(1M).

NAME
rquotad - remote quota server

SYNOPSIS
/usr/sbin/rpc.rquotad

DESCRIPTION
rquotad is an RPC server that returns quotas for a user of a local file system currently mounted by a remote machine by means of NFS (see *rpc*(3C)). The results are used by **quota** to display user quotas for remote file systems (see *quota*(1)). **rquotad** is normally invoked by **inetd** (see *inetd*(1M)).

AUTHOR
Disk Quotas were developed by the University of California, Berkeley, Sun Microsystems, Inc., and HP.

FILES
directory/**quotas**	Quota statistics static storage for a file system, where *directory* is the root of the file system.

SEE ALSO
inetd(1M), rpc(3C), services(4), quota(5), nfs(7).

NAME

rstatd - kernel statistics server

SYNOPSIS

/usr/lib/netsvc/rstat/rpc.rstatd [**-l** *log_file*] [**-e**|**-n**]

DESCRIPTION

rstatd is an RPC server that returns performance statistics obtained from the kernel. The **rup** utility prints this information (see *rup*(1)).

inetd invokes **rstatd** through **/etc/inetd.conf** (see *inetd*(1M)).

Options

rstatd recognizes the following options and command-line arguments:

-l *log_file* Log any errors to the named log file, *log_file*. Errors are not logged if the **-l** option is not specified.

Information logged to the file includes date and time of the error, the host name, process ID and name of the function generating the error, and the error message. Note that different services can share a single log file because enough information is included to uniquely identify each error.

-e Exit after serving each RPC request. Using the **-e** option, the **inetd** security file **/var/adm/inetd.sec** can control access to RPC services.

-n Exit only if

- **portmap** dies (see *portmap*(1M)),
- another **rpc.rstatd** registers with **portmap**, or
- **rpc.rstatd** becomes unregistered with *portmap*.

The **-n** option is more efficient since a new process is not launched for each RPC request. Note, this option is the default.

AUTHOR

rstatd was developed by Sun Microsystems, Inc.

SEE ALSO

rup(1), inetd(1M), portmap(1M), inetd.conf(4), inetd.sec(4), services(4).

NAME

runacct - run daily accounting

SYNOPSIS

/usr/sbin/acct/runacct [*mmdd* [*state*]]

DESCRIPTION

runacct is the main daily accounting shell procedure. It is normally initiated via *cron*(1M). *runacct* processes connect, fee, disk, and process accounting files. It also prepares summary files for *prdaily* or billing purposes.

runacct takes care not to damage active accounting files or summary files in the event of errors. It records its progress by writing descriptive diagnostic messages into **active**. When an error is detected, a message is written to **/dev/console**, mail (see *mail*(1), *mailx*(1), or *elm*(1)) is sent to **root** and **adm**, and *runacct* terminates. *runacct* uses a series of lock files to protect against re-invocation. The files **lock** and **lock1** are used to prevent simultaneous invocation, and **lastdate** is used to prevent more than one invocation per day.

runacct breaks its processing into separate, restartable *states* using **statefile** to remember the last *state* completed. It accomplishes this by writing the *state* name into **statefile**. *runacct* then looks in **statefile** to see what it has done and to determine what to process next. *states* are executed in the following order:

SETUP	Move active accounting files into working files.
WTMPFIX	Verify integrity of **wtmp** file, correcting date changes if necessary.
CONNECT	Produce connect session records in **tacct.h** format.
PROCESS	Convert process accounting records into **tacct.h** format.
MERGE	Merge the connect and process accounting records.
FEES	Convert output of *chargefee* into **tacct.h** format and merge with connect and process accounting records.
DISK	Merge disk accounting records with connect, process, and fee accounting records.
MERGETACCT	Merge the daily total accounting records in **daytacct** with the summary total accounting records in **/var/adm/acct/sum/tacct**.
CMS	Produce command summaries.
USEREXIT	Any installation-dependent accounting programs can be included here.
CLEANUP	Cleanup temporary files and exit.

To restart *runacct* after a failure, first check the **active** file for diagnostics, then fix up any corrupted data files such as **pacct** or **wtmp**. The **lock** files and **lastdate** file must be removed before *runacct* can be restarted. The argument *mmdd* is necessary if *runacct* is being restarted, and specifies the month and day for which *runacct* will rerun the accounting. Entry point for processing is based on the contents of **statefile**; to override this, include the desired *state* on the command line to designate where processing should begin.

EXAMPLES

To start *runacct*.

```
nohup runacct 2> /var/adm/acct/nite/fd2log &
```

To restart *runacct*.

```
nohup runacct 0601 2>> /var/adm/acct/nite/fd2log &
```

To restart *runacct* at a specific *state*.

```
nohup runacct 0601 MERGE 2>> /var/adm/acct/nite/fd2log &
```

WARNINGS

Normally it is not a good idea to restart *runacct* in its **SETUP** *state*. Run **SETUP** manually, then restart via:

runacct *mmdd* **WTMPFIX**

If *runacct* failed in its **PROCESS** *state*, remove the last **ptacct** file because it will not be complete.

FILES

/var/adm/acct/nite/active
/var/adm/acct/nite/daytacct
/var/adm/acct/nite/lastdate
/var/adm/acct/nite/lock
/var/adm/acct/nite/lock1
/var/adm/pacct∗
/var/adm/acct/nite/ptacct∗**.***mmdd*
/var/adm/acct/nite/statefile
/var/adm/wtmp

SEE ALSO

mail(1), acct(1M), acctcms(1M), acctcom(1M), acctcon(1M), acctmerg(1M), acctprc(1M), acctsh(1M), cron(1M), fwtmp(1M), acct(2), acct(4), utmp(4).

STANDARDS CONFORMANCE

runacct: SVID2, SVID3

NAME

rusersd - network username server

SYNOPSIS

/usr/lib/netsvc/rusers/rpc.rusersd [**-l** *log_file*] [**-e**|**-n**]

DESCRIPTION

rusersd is an RPC server that returns a list of users on the network. The **rusers** command prints this information (see *rusers*(1)).

inetd invokes **rusersd** through **/etc/inetd.conf** (see *inetd*(1M)).

Options

rusersd recognizes the following options and command-line arguments:

-l *log_file* Log any errors to the named log file, *log_file*. Errors are not logged if the **-l** option is not specified.

Information logged to the file includes date and time of the error, the host name, process ID and name of the function generating the error, and the error message. Note that different services can share a single log file since enough information is included to uniquely identify each error.

-e Exit after serving each RPC request. Using the **-e** option, the **inetd** security file **/var/adm/inetd.sec** can control access to RPC services.

-n Exit only if

- **portmap** dies (see *portmap*(1M)),
- another **rpc.rusersd** registers with **portmap**, or
- **rpc.rusersd** becomes unregistered with **portmap**.

The **-n** option is more efficient because a new process is not launched for each RPC request. This option is the default.

AUTHOR

rusersd was developed by Sun Microsystems, Inc.

SEE ALSO

rusers(1), inetd(1M), portmap(1M), inetd.conf(4), inetd.sec(4), services(4).

NAME

rwall - write to all users over a network

SYNOPSIS

/usr/sbin/rwall *hostname* ...

/usr/sbin/rwall -n *netgroup* ...

/usr/sbin/rwall -h *host* **-n** *netgroup*

DESCRIPTION

rwall reads a message from standard input until EOF, then sends the message, preceded by the line **Broadcast Message** ..., to all users logged in on the specified host machines. With the **-n** option, **rwall** sends the message to the specified network hosts defined in **/etc/netgroup** (see *netgroup*(4)).

A machine can only receive such a message if it is running **rwalld**, which is normally started from **/etc/inetd.conf** by the **inetd** daemon (see *inetd*(1M)).

WARNINGS

The timeout is kept fairly short so that the message can be sent to a large group of machines (some of which may be down) in a reasonable amount of time. Thus, the message may not get through to a heavily loaded machine.

AUTHOR

rwall was developed by Sun Microsystems, Inc.

FILES

/etc/inetd.conf

SEE ALSO

rwalld(1M), shutdown(1M), wall(1M), netgroup(4).

NAME

rwalld - network rwall server

SYNOPSIS

/usr/lib/netsvc/rwall/rpc.rwalld [**-l** *log_file*] [**-e**|**-n**]

DESCRIPTION

rwalld is an RPC server that handles **rwall** requests (see *rwall*(1)). **rwalld** calls **wall** to send a message to all users logged into the host on which **rwalld** is running (see *wall*(1)).

inetd invokes **rwalld** through **/etc/inetd.conf** (see *inetd*(1M)).

Options

rwalld recognizes the following options and command-line options:

-l *log_file* Log any errors to *log_file*. Errors are not logged if the **-l** option is not specified.

Information logged to the log file includes date and time of the error, the host name, process ID and name of the function generating the error, and the error message. Note that different services can share a single log file because enough information is included to uniquely identify each error.

-e Exit after serving each RPC request. Using the **-e** option, the **inetd** security file **/var/adm/inetd.sec** can control access to RPC services.

-n Exit only if:

- **portmap** dies (see *portmap*(1M)),
- another **rpc.rwalld** registers with **portmap**, or
- **rpc.rwalld** becomes unregistered with **portmap**.

The **-n** option is more efficient because a new process is not launched for each RPC request. Note, this option is the default.

AUTHOR

rwalld was developed by Sun Microsystems, Inc.

SEE ALSO

inetd(1M), portmap(1M), rwall(1M), wall(1M), inetd.conf(4), inetd.sec(4), services(4).

NAME

rwhod - system status server

SYNOPSIS

/usr/lbin/rwhod [**-s**] [**-r**]

DESCRIPTION

rwhod is the server that maintains the database used by **rwho** and **ruptime** (see *rwho*(1) and *ruptime*(1)). **rwhod** sends status information to and receives status information from other nodes on the local network that are running **rwhod**.

rwhod is started at system boot time if the RWHOD variable is set to 1 in the file **/etc/rc.config.d/netdaemons**.

As an information sender, it periodically queries the state of the system and constructs status messages that are broadcast on a network.

As an information receiver, it listens for other **rwhod** servers' status messages, validates them, then records them in a collection of files located in the **/var/spool/rwho** directory.

By default, **rwhod** both sends and receives information. **rwhod** also supports the following options:

-s Configures server to be an information sender only.

-r Configures server to be an information receiver only.

Status messages are generated approximately once every three minutes. **rwhod** transmits and receives messages at the port indicated in the **who** service specification (see *services*(4)). The messages sent and received, are of the form:

```
struct  outmp {
        char    out_line[8];            /* tty name */
        char    out_name[8];            /* user id */
        long    out_time;               /* time on */
};

struct  whod {
        char    wd_vers;
        char    wd_type;
        char    wd_fill[2];
        int     wd_sendtime;
        int     wd_recvtime;
        char    wd_hostname[32];
        int     wd_loadav[3];
        int     wd_boottime;
        struct  whoent {
                struct  outmp we_utmp;
                int     we_idle;
        } wd_we[1024 / sizeof (struct whoent)];
};
```

All fields are converted to network byte order before transmission. System load averages are calculated from the number of jobs in the run queue over the last 1-, 5- and 15-minute intervals. The host name included is the one returned by the **gethostname()** system call (see *gethostname*(2)). The array at the end of the message contains information about the users logged in on the sending machine. This information includes the contents of the **utmp** entry for each non-idle terminal line and a value indicating the time since a character was last received on the terminal line (see *utmp*(4)).

rwhod discards received messages if they did **not** originate at a **rwho** server's port, or if the host's name, as specified in the message, contains any unprintable ASCII characters.

Valid messages received by **rwhod** are placed in files named **whod.***hostname* in the **/var/spool/rwho** directory. These files contain only the most recent message in the format described above.

WARNINGS

rwhod does not relay status information between networks. Users often incorrectly interpret the server dying as a machine going down.

AUTHOR

rwhod was developed by the University of California, Berkeley.

FILES

/var/spool/rwho/whod.*	Information about other machines.

SEE ALSO

rwho(1), ruptime(1).

NAME

sa1, sa2, sadc - system activity report package

SYNOPSIS

/usr/lbin/sa/sa1 [*t n*]

/usr/lbin/sa/sa2 [**-ubdycwaqvmA**] [**-s** *time*] [**-e** *time*] [**-i** *sec*]

/usr/lbin/sa/sadc [*t n*] [*ofile*]

DESCRIPTION

System activity data can be accessed at the special request of a user (see *sar*(1)) and automatically on a routine basis as described here. The operating system contains a number of counters that are incremented as various system actions occur. These include CPU utilization counters, buffer usage counters, disk and tape I/O activity counters, tty device activity counters, switching and system-call counters, file-access counters, queue activity counters, and counters for inter-process communications.

sadc and shell procedures **sa1** and **sa2** are used to sample, save, and process this data.

sadc, the data collector, samples system data *n* times every *t* seconds and writes in binary format to *ofile* or to standard output. If *t* and *n* are omitted, a special record is written. This facility is used at system boot time to mark the time at which the counters restart from zero. Executing the following command in a system startup script:

```
/usr/lbin/sa/sadc /var/adm/sa/sa‘date +%d‘
```

writes the special record to the daily data file to mark the system restart. Instructions for creating system startup scripts may be found in the 10.0 File System Layout White Paper, which is online in file **/usr/share/doc/filesys.ps**.

The shell script **sa1**, a variant of **sadc**, is used to collect and store data in binary file **/var/adm/sa/sa***dd* where *dd* is the current day. The arguments *t* and *n* cause records to be written *n* times at an interval of *t* seconds, or once if omitted. The following entries, if placed in **crontab**, produce records every 20 minutes during working hours and hourly otherwise (see *cron*(1M)):

```
0 *    * * 0,6  /usr/lbin/sa/sa1
0 8-17 * * 1-5  /usr/lbin/sa/sa1 1200 3
0 18-7 * * 1-5  /usr/lbin/sa/sa1
```

The shell script **sa2**, a variant of **sar**, writes a daily report in file **/var/adm/sa/sar***dd*. The options are explained in *sar*(1). The following **crontab** entry reports important activities hourly during the working day:

```
5 18 * * 1-5  /usr/lbin/sa/sa2 -s 8:00 -e 18:01 -i 3600 -A
```

The structure of the binary daily data file is:

```
struct sa {
    struct sysinfo si;      /* see /usr/include/sys/sysinfo.h  */
    int  sztext;            /* current entries of text table   */
    int  szinode;           /* current entries of inode table  */
    int  szfile;            /* current entries of file table   */
    int  szproc;            /* current entries of proc table   */
    int  msztext;           /* size of text table  */
    int  mszinode;          /* size of inode table */
    int  mszfile;           /* size of file table  */
    int  mszproc;           /* size of proc table  */
    long  textovf;          /* cumul. overflows of text table  */
    long  inodeovf;         /* cumul. overflows of inode table */
    long  fileovf;          /* cumul. overflows of file table  */
    long  procovf;          /* cumul. overflows of proc table  */
    time_t  ts;             /* time stamp, seconds  */
    long  devio[NDEVS][4];  /* device info for up to NDEVS units */
#define IO_OPS  0           /* cumul. I/O requests  */
#define IO_BCNT 1           /* cumul. blocks transferred */
#define IO_ACT  2           /* cumul. drive busy time in ticks  */
#define IO_RESP 3           /* cumul. I/O resp time in ticks  */
};
```

S

FILES

/tmp/sa.*adrfl*	address file
/var/adm/sa/sa*dd*	daily data file
/var/adm/sa/sar*dd*	daily report file

SEE ALSO

cron(1M), sar(1), timex(1).

STANDARDS CONFORMANCE

sa1: SVID2, SVID3

sa2: SVID2, SVID3

sadc: SVID2, SVID3

NAME

sam - system administration manager

SYNOPSIS

/usr/sbin/sam [**-display** *display*] [**-f** *login*] [**-r**]

DESCRIPTION

The **sam** command starts a menu-driven System Administration Manager program (SAM) for performing system administration tasks with only limited, specialized knowledge of the HP-UX operating system. SAM discovers most aspects of a system's configuration through automated inquiries and tests. Help menus describe how to use SAM and perform the various management tasks. Press the **F1** function key for help on a currently highlighted field and for more information not covered in this manpage. Status messages and a log file monitor keep the user informed of what SAM is doing.

Running SAM

SAM has been tuned to run in the Motif environment, but it can be run on text terminals as well. To run SAM in the Motif environment, be sure that Motif has been installed on your system, and that the **DISPLAY** environment variable is set to the system name on which the SAM screens should be displayed (or use the **-display** command line option).

Generally, SAM requires superuser (user **root**) privileges to execute successfully. However, SAM can be configured (through the use of "Restricted SAM"; see below) to allow subsets of its capabilities to be used by non-**root** users. When Restricted SAM is used, non-**root** users are promoted to **root** when necessary to enable them to execute successfully.

Options

SAM recognizes the following options.

-display *display*	Set the **DISPLAY** value for the duration of the SAM session.
-f *login*	Execute SAM with the privileges associated with the specified *login*. When used in conjunction with **-r**, the Restricted SAM Builder is invoked and initialized with the privileges associated with the specified *login*. You must be a superuser to use this option. See "Restricted SAM" below for more information.
-r	Invoke the Restricted SAM Builder. This enables the system administrator to provide limited non-superuser access to SAM functionality. You must be a superuser to use this option. See "Restricted SAM" below for more information.

SAM Functional Areas

SAM performs these system administration tasks:

Auditing and Security (Trusted Systems)

- Set global system security policies - Add, modify and remove commands from the list of Authenticated commands.
- Turn the Auditing system on or off.
- Set the parameters for the Audit Logs and Size Monitor.
- View all or selected parts of the audit logs.
- Modify (or view) which users, events, and/or system calls get audited.
- Convert your system to a Trusted System.
- Convert your system to a non-Trusted System.

Backup and Recovery

- Interactively back up files to a valid backup device (cartridge tape, cartridge tape autochanger, magnetic tape, DAT, magneto-optical disk, or magneto-optical disk autochanger). The SAM interface is suspended so that you can read and/or respond to the interactive messages produced by **fbackup** (see *fbackup*(1M)).
- Recover files online from a valid backup device. The SAM interface is suspended so that you can read/respond to the interactive messages produced by **frecover** (see *frecover*(1M)).
- Add to, delete from, or view the automated backup schedule.
- Obtain a list of files from a backup tape.
- View various backup and recovery log files.

Disk and File Systems Management

- Add, configure, or unconfigure disk devices, including hard drives, floppy drives, CD-ROMs, magneto-optical devices and disk arrays.
- Add, modify, or remove local file systems, or convert them to long file names.
- Configure HFS or VxFS file systems.
- Remote (NFS) file systems configuration, including:
  - Add, modify, or remove remote (NFS) file systems.
  - Allow or disallow access by remote systems to local file systems.
  - Modify RPC (Remote Procedure Call) services' security.
- Add, remove, or modify device or file system swap.
- Change the primary swap device.
- Add, modify, or remove dump devices.
- Examine, create, extend, or reduce a volume-group pool of disks.
- Create, extend or change number of mirrored copies of a logical volume and associated file system.
- Remove a logical volume or increase its size.
- Split or merge mirrored copies of a logical volume.
- Share or unshare volume groups (only on ServiceGuard clusters running MC/LockManager distributed lock-manager software).

Kernel and Device Configuration

- Add/remove static drivers and DLKM modules to/from a kernel.
- Modify static and dynamic tunable parameter values in the kernel.
- Modify dump device configuration in the kernel.
- Add or remove optional subsystems such as NFS, LAN, NS, CD-ROM, etc.
- Generate a new kernel.

Networks/Communications

- Configure one or more LAN cards.
- Configure ARPA services.
- Configure the Network File System (NFS).
- Configure X.25 card or cards and PAD (Packet Assembler/Disassembler) services (if X.25 has been purchased).

Peripheral Devices Management

- Administer the LP spooler or Distributed Print Services and associated printers and plotters (see "Printer and Plotter Management" below).
- Add, modify, or remove the configuration of disk devices.

- Add or remove terminals and modems.
- Configure terminal security policies (Trusted Systems only).
- Lock and unlock terminals (Trusted Systems only).
- Add or remove tape drives.
- Add or remove hardware interface cards.
- View current configuration of peripherals and disk space information.

Printer and Plotter Management
SAM supports two methods for managing printers and plotters:

- **LP Spooler -** Manage local, remote, and networked printers and plotters.
- **HP Distributed Print Service (HPDPS) -** Manage physical printers (parallel, serial, or network interface and remote printers), logical printers, print queues, spoolers, and supervisors.

Process Management

- Kill, stop or continue processes.
- Change the nice priority of processes.
- View the current status of processes.
- Schedule periodic tasks via cron.
- View current periodic (cron) tasks.
- Run performance monitors.
- Display system properties such as: machine model and ID; number of installed processors, their version and speed; operating-system release version; swap statistics, real, physical, and virtual memory statistics; network connection information.

Remote Administration

- Configure remote systems for remote administration.
- Execute SAM on systems configured for remote administration.

Routine Tasks

- Shut down the system.
- View and remove large files. Specify size and time-since-accessed of large files to display or remove.
- View and remove unowned files. Specify size and time-since-accessed of unowned files to display or remove.
- View and remove core files.
- View and trim ASCII or non-ASCII log files. Add or remove files from the list of files to monitor. Set recommended size for trimming.

User and Group Account Management

- Add, remove, view, and modify user accounts.
- Modify a user account's group membership.
- Set up password aging for a user account.
- Add, remove, view, and modify groups.
- Deactivate and reactivate user accounts.
- Manage trusted system security policies on a per-user basis.

Adding New Functionality to SAM

You can easily add stand-alone commands, programs, and scripts to SAM. SAM is suspended while the executable program is running. When it finishes, the SAM interface is restored. You can also write your own help screen for each menu item you create. To add functionality to SAM, select the "Add Custom Menu Item" or "Add Custom Menu Group" action items from the SAM Areas menu. (Note that the new item is added to the hierarchy that is currently displayed, so you need to navigate to the desired hierarchy before adding the item.)

Restricted SAM

SAM can be configured to provide a subset of its functionality to certain users or groups of users. It can also be used to build a template file for assigning SAM access restrictions on multiple systems. This is done through the Restricted SAM Builder. System administrators access the Restricted SAM Builder by invoking SAM with the **-r** option (see "Options" above). In the Builder, system administrators may assign subsets of SAM functionality on a per-user or per-group basis. Once set up, the **-f** option (see "Options" above) can then be used by system administrators to verify that the appropriate SAM functional areas, and only those areas, are available to the specified user.

A nonroot user that has been given Restricted SAM privileges simply executes **/usr/sbin/sam** and sees only those areas the user is privileged to access. For security reasons, the "List" and "Shell Escape" choices are not provided. (Note that some SAM functional areas require the user to be promoted to root in order to execute successfully. SAM does this automatically as needed.)

SAM provides a default set of SAM functional areas that the system administrator can assign to other users. Of course, system administrators are able to assign custom lists of SAM functional areas to users as necessary.

SAM Logging

All actions taken by SAM are logged into the SAM log file **/var/sam/log/samlog**. The log entries in this file can be viewed via the SAM utility **samlog_viewer** (see *samlog_viewer*(1)). **samlog_viewer** can filter the log file by user name, by time of log entry creation, and by level of detail.

The "Options" menu in the SAM Areas Menu enables you to start a log file viewer and to control certain logging options. These options include whether or not SAM should automatically start a log file viewer whenever SAM is executed, whether or not SAM should trim the log file automatically, and what maximum log file size should be enforced if automatic log file trimming is selected.

VT320 Terminal Support

Because the VT320 terminal has predefined local functions for keys labeled as **F1**, **F2**, **F3** and **F4**, users should use following mapping when they desire to use function keys:

HP or Wyse60	**VT320 or HP 700/60 in VT320 mode**
F1	**PF2** *(1)*
F2	**PF1** *(1)*
F3	**spacebar**
F4	**PF3** *(1)*
F5	**F10**, [EXIT], **F5** *(2)*
F6	none
F7	**F18**, first unlabeled key to right of **Pause/Break** *(2)*
F8	**F19**, second unlabeled key to right of **Pause/Break** *(2)*

(1) See the "Configuration: HP 700/60 in DEC mode, or DEC terminals with PC-AT-type keyboard" subsection below.

(2) When using PC-AT keyboard with HP 700/60 in VT320 mode.

Since DEC terminals do not support the softkey menu, that menu is not displayed on those terminals.

Many applications use **TAB** for forward navigation (moving from one field to another) and **shift-TAB** for backward navigation. Users having DEC terminals or using terminals in DEC emulation modes such as VT100 or VT320 may note that these terminals/emulators may produce the same character for **TAB** and **shift-TAB**. As such, it is impossible for an application to distinguish between the two and both of them are treated as if the **TAB** key was pressed. This presents an inconvenience to users if they want to go backward. In most cases, they should complete rest of the input fields and get back to the desired field later.

VT100 Terminal Support

VT100 does not allow the **F1**–**F8** function keys to be configured. Therefore, the following keyboard mappings apply to VT100 terminals:

HP or Wyse60	**VT100 or HP 700/60 in VT100 mode**
F1	**PF2** *(1)*

F2	**PF1** *(1)*
F3	**spacebar**
F4	**PF3**, **spacebar** or **PF3**, **=** *(1)*
F5	**Return**
F6	none
F7	none
F8	none

(1) See the "Configuration: HP 700/60 in DEC mode, or DEC terminals with PC-AT-type keyboard" subsection below.

See the comments on softkeys and **TAB** keys in the "VT320 Terminal Support" subsection above.

Configuration: HP 700/60 Terminal in DEC Mode, or DEC Terminal with PC-AT-Type Keyboard

Customers using the following configuration may want to be aware of the following keyboard difference.

It may be possible for a user with the "HP 700/60 terminal in DEC mode, or DEC terminal with PC-AT-type keyboard" configuration to be told to press function key **F1** through **F4** to achieve some desired result. For an HP 700/60 terminal in DEC mode or DEC terminals, these functions keys may be mapped onto **PF1**–**PF4** keys. However, the PC-AT-type keyboard does not provide **PF1**–**PF4** keys, as does the DEC/ANSI keyboard.

Key	**Maps to**
Num Lock	**PF1**
/	**PF2**
*****	**PF3**
–	**PF4**

These keys are above the number pad on the right side of the keyboard. Please note that although this keyboard is called a PC AT-type keyboard, it is supplied by HP. A PC AT-type keyboard can be recognized by location of Esc key at the left-top of the keyboard.

Wyse60 Terminal Support

On Wyse60, use the **DEL** key (located next to **Backspace**) to backspace. On an HP 700/60 with a PC AT-type keyboard in Wyse60 mode, the **DEL** key is located in the bottom row on the number pad.

Wyse60 terminals provide a single line to display softkey labels unlike HP terminals which provide two lines. Sometimes this may result in truncated softkey labels. For example, the `Help on Context` label for **F1** may appear as `Help on C`. Some standard labels for screen-oriented applications, such as SAM and `swinstall` are as follows:

The SAM label:	**May appear on the Wyse60 as:**
`Help On Context`	`Help On C`
`Select/Deselect`	`Select/D`
`Menubar on/off`	`Menubar`

DEPENDENCIES

SAM runs in an X Window environment as well as on the following kinds of terminals or terminal emulators:

- HP-compatible terminal with programmable function keys and on-screen display of function key labels.
- VT-100 and VT-320
- WY30 and WY60

Depending on what other applications are running concurrently with SAM, more swap space may be required. SAM requires the following amounts of internal memory:

8 MB	If using terminal based version of SAM.
16 MB	If using Motif X Window version of SAM.

For more detailed information about how to use SAM on a terminal, see the *Managing Systems and Workgroups* manual.

AUTHOR

sam was developed by HP.

FILES

/etc/sam/custom	Directory where SAM stores user privileges.
/etc/sam/rmfiles.excl	File containing a list of files and directories that are excluded from removal by SAM.
/etc/sam/rmuser.excl	File containing a list of users that are excluded from removal by SAM.
/usr/sam/bin	Directory containing executable files, which can be used outside of any SAM session.
/usr/sam/help/$LANG	Directory containing SAM language specific online help files.
/usr/sam/lbin	Directory containing SAM executables, which are intended only for use by SAM and are not supported in any other context.
/usr/sam/lib	Directory for internal configuration files.
/var/sam	Directory for working space, including lock files (if a SAM session dies, it may leave behind a spurious lock file), preferences, logging, and temporary files.
/var/sam/log/samlog	File containing unformatted SAM logging messages. This file should not be modified by users. Use **samlog_viewer** to view the contents of this file (see *samlog_viewer*(1)).
/var/sam/log/samlog.old	Previous SAM log file. This file is created by SAM when **/var/sam/log/samlog** is larger than the user specified limit. Use **samlog_viewer** with its **-f** option to view the contents of this file (see *samlog_viewer*(1)).

SEE ALSO

samlog_viewer(1), parmgr(1M).

These manuals are available on the Web at **docs.hp.com**:

- *Managing Systems and Workgroups*
- *Installing and Administering Internet Services*
- *Installing and Administering LAN/9000*
- *Installing and Administering NFS Services*
- *X.25/9000 User's Manual*

NAME

sar - system activity reporter

SYNOPSIS

sar [**-ubdycwaqvmAMS**] [**-o** *file*] *t* [*n*]

sar [**-ubdycwaqvmAMS**] [**-s** *time*] [**-e** *time*] [**-i** *sec*] [**-f** *file*]

DESCRIPTION

In the first form above, **sar** samples cumulative activity counters in the operating system at *n* intervals of *t* seconds. If the **-o** option is specified, it saves the samples in *file* in binary format. The default value of *n* is 1. In the second form, with no sampling interval specified, **sar** extracts data from a previously recorded *file*, either the one specified by **-f** option or, by default, the standard system activity daily data file **/var/adm/sa/sa***dd* for the current day *dd*. The starting and ending times of the report can be bounded via the **-s** and **-e** *time* arguments of the form *hh*[:*mm*[:*ss*]]. The **-i** option selects records at *sec*-second intervals. Otherwise, all intervals found in the data file are reported.

In either case, subsets of data to be printed are specified by option:

-u Report CPU utilization (the default); portion of time running in one of several modes. On a multi-processor system, if the **-M** option is used together with the **-u** option, per-CPU utilization as well as the average CPU utilization of all the active processors are reported. If the **-M** option is not used, only the average CPU utilization of all the active processors is reported:

cpu	cpu number (only on a multi-processor system with the **-M** option);
%usr	user mode;
%sys	system mode;
%wio	idle with some process waiting for I/O (only block I/O, raw I/O, or VM pageins/swapins indicated);
%idle	otherwise idle.

-b Report buffer activity:

bread/s	Number of physical reads per second from the disk (or other block devices) to the buffer cache;
bwrit/s	Number of physical writes per second from the buffer cache to the disk (or other block device);
lread/s	Number of reads per second from buffer cache;
lwrit/s	Number of writes per second to buffer cache;
%rcache	Buffer cache hit ratio for read requests e.g., 1 − bread/lread;
%wcache	Buffer cache hit ratio for write requests e.g., 1 − bwrit/lwrit;
pread/s	Number of reads per second from character device using the **physio()** (raw I/O) mechanism;
pwrit/s	Number of writes per second to character device using the **physio()** (i.e., raw I/O) mechanism; mechanism.

-d Report activity for each block device, e.g., disk or tape drive. One line is printed for each device that had activity during the last interval. If no devices were active, a blank line is printed. Each line contains the following data:

device	Logical name of the device and its corresponding instance. Devices are categorized into the following device types: disk3 – SCSI and NIO FL disks sdisk – SCSI disks;
%busy	Portion of time device was busy servicing a request;
avque	Average number of requests outstanding for the device;
r+w/s	Number of data transfers per second (read and writes) from and to the device;

S

blks/s	Number of bytes transferred (in 512-byte units) from and to the device;
avwait	Average time (in milliseconds) that transfer requests waited idly on queue for the device;
avserv	Average time (in milliseconds) to service each transfer request (includes seek, rotational latency, and data transfer times) for the device.

-y Report tty device activity:

rawch/s	Raw input characters per second;
canch/s	Input characters per second processed by **canon()**;
outch/s	Output characters per second;
rcvin/s	Receive incoming character interrupts per second;
xmtin/s	Transmit outgoing character interrupts per second;
mdmin/s	Modem interrupt rate (not supported; always 0).

-c Report system calls:

scall/s	Number of system calls of all types per second;
sread/s	Number of **read()** and/or **readv()** system calls per second;
swrit/s	Number of **write()** and/or **writev()** system calls per second;
fork/s	Number of **fork()** and/or **vfork()** system calls per second;
exec/s	Number of **exec()** system calls per second;
rchar/s	Number of characters transferred by read system calls block devices only) per second;
wchar/s	Number of characters transferred by write system calls (block devices only) per second.

-w Report system swapping and switching activity:

swpin/s	Number of process swapins per second;
swpot/s	Number of process swapouts per second;
bswin/s	Number of 512-byte units transferred for swapins per second;
bswot/s	Number of 512-byte units transferred for swapouts per second;
pswch/s	Number of process context switches per second.

-a Report use of file access system routines:

iget/s	Number of file system **iget()** calls per second;
namei/s	Number of file system **lookuppn()** (pathname translation) calls per second;
dirblk/s	Number of file system blocks read per second doing directory lookup.

-q Report average queue length while occupied, and percent of time occupied. On a multiprocessor machine, if the **-M** option is used together with the **-q** option, the per-CPU run queue as well as the average run queue of all the active processors are reported. If the **-M** option is not used, only the average run queue information of all the active processors is reported:

cpu	cpu number (only on a multi-processor system and used with the **-M** option)
runq-sz	Average length of the run queue(s) of processes (in memory and runnable);
%runocc	The percentage of time the run queue(s) were occupied by processes (in memory and runnable);
swpq-sz	Average length of the swap queue of runnable processes (processes swapped out but ready to run);

%swpocc	The percentage of time the swap queue of runnable processes (processes swapped out but ready to run) was occupied.

-v Report status of text, process, inode and file tables:

text-sz	(Not Applicable);
proc-sz	The current-size and maximum-size of the process table;
inod-sz	The current-size and maximum-size of the inode table (inode cache);
file-sz	The current-size and maximum-size of the system file table;
text-ov	(Not Applicable);
proc-ov	The number of times the process table overflowed (number of times the kernel could not find any available process table entries) between sample points;
inod-ov	The number of times the inode table (inode cache) overflowed (number of times the kernel could not find any available inode table entries) between sample points;
file-ov	The number of times the system file table overflowed (number of times the kernel could not find any available file table entries) between sample points.

-m Report message and semaphore activities:

msg/s	Number of System V **msgrcv()** calls per second;
sema/s	Number of System V **semop()** calls per second;
select/s	Number of System V **select()** calls per second. This value will only be reported if the "-S" option is also explicitly specified.

-A Report all data. Equivalent to **-udqbwcayvm**.

-M Report the per-processor data on a multi-processor system when used with **-q** and/or **-u** options. If the **-M** option is not used on a multi-processor system, the output format of the **-u** and **-q** options is the same as the uni-processor output format and the data reported is the average value of all the active processors.

EXAMPLES

Watch CPU activity evolve for 5 seconds:

```
sar 1 5
```

Watch CPU activity evolve for 10 minutes and save data:

```
sar -o temp 60 10
```

Review disk and tape activity from that period later:

```
sar -d -f temp
```

Review cpu utilization on a multi-processor system later:

```
sar -u -M  -f temp
```

WARNINGS

Users of **sar** must not rely on the exact field widths and spacing of its output, as these will vary depending on the system, the release of HP-UX, and the data to be displayed.

FILES

/var/adm/sa/sa*dd* daily data file, where *dd* is two digits representing the day of the month.

SEE ALSO

sa1(1M).

STANDARDS CONFORMANCE

sar: SVID2, SVID3

NAME

savecrash - save a crash dump of the operating system

SYNOPSIS

/sbin/savecrash [**-cflprvzZ**] [**-D** *dumpdevice* **-O** *offset*] [**-d** *sysfile*]

[**-m** *minfree*] [**-s** *chunksize*] [**-t** *tapedevice*]

[**-w** **NOSWAP**|**SWAPEACH**|**SWAPEND**] [*dirname*]

DESCRIPTION

savecrash saves the crash dump information of the system (assuming one was made when the system crashed) and writes a reboot message in the shutdown log file.

dirname is the name of the existing directory in which to store the crash dump; the default is **/var/adm/crash**.

savecrash saves the crash image and related files in the directory *dirname*/**crash.***n*. The trailing *n* in the directory name is a number that increases by one every time **savecrash** is run with the same *dirname*. This number is kept in the file *dirname*/**bounds**, which is created if it does not already exist.

Usually, **savecrash** creates the **INDEX** file in the crash directory from the crash dump header, copies all kernel modules that were loaded in memory at the time of the crash, and copies all dump device contents into crash image files.

When **savecrash** writes out a crash dump directory, it checks the space available on the file system containing *dirname*. **savecrash** will not use that portion of the file system space which is reserved for the superuser. Additional space on the file system can be reserved for other uses with **-m** *minfree*, where *minfree* is the amount of additional space to reserve. This option is useful for ensuring enough file system space for normal system activities after a panic.

If there is insufficient space in the file system for the portions of the crash dump that need to be saved, **savecrash** will save as much as will fit in the available space. (Priority is given to the index file, then to the kernel module files, and then to the physical memory image.) The dump will be considered saved, and **savecrash** will not attempt to save it again, unless there was insufficient space for *any* of the physical memory image. (See the description of option **-r**.)

savecrash also writes a reboot message in the shutdown log file (**/etc/shutdownlog**), if one exists. (If a shutdown log file does not exist, **savecrash** does not create one.) If the system crashes as a result of a kernel panic, **savecrash** also records the panic string in the shutdown log.

By default, when the primary paging device is not used as one of the dump devices or after the crash image on the primary paging device has been saved, **savecrash** runs in the background. This reduces system boot-up time by allowing the system to be run with only the primary paging device.

S

It is possible for dump devices to be used also as paging devices. If **savecrash** determines that a dump device is already enabled for paging, and that paging activity has already taken place on that device, a warning message will indicate that the dump may be invalid. If a dump device has not already been enabled for paging, **savecrash** prevents paging from being enabled to the device by creating the file **/etc/savecore.LCK**. **swapon** does not enable the device for paging if the device is locked in **/etc/savecore.LCK** (see *swapon*(1M) for more details). As **savecrash** finishes saving the image from each dump device, it updates the **/etc/savecore.LCK** file and optionally executes **swapon** to enable paging on the device.

Options

-c Mark the dump in the dump device as saved, without performing any other action. The **-c** option is useful for manually inhibiting dump actions called by **/sbin/init.d/savecrash**.

-f Run **savecrash** in the foreground only. By default, **savecrash** runs in the background when the primary paging device does not contain an unsaved portion of the crash image. Turning this option on increases system boot-up time, but guarantees that the dump has been saved when control returns to the caller.

-l Logs the panic information to **/etc/shutdownlog** as described above, but does not actually save the dump. The dump is marked as saved so that future invocations of **savecrash** do not create duplicate log entries.

-p Only preserves swap-endangered dump device contents into crash image files. Swap-endangered dump devices are those devices that are also configured as swap devices by the system. If all dump devices are configured as swap devices, the entire dump will be preserved in the crash directory. If no swap devices are used as dump devices (dedicated dump devices), only the **INDEX** file and kernel modules will be copied into the crash directory.

-r Resaves a dump that a previous invocation of **savecrash** has marked as already saved. This is useful if the first invocation did ran out of space, and enough space has since been freed to try again.

-v Enables additional progress messages and diagnostics.

-z **savecrash** will compress all physical memory image files and kernel module files in the dump directory.

-Z **savecrash** will not compress any files in the dump directory.

If neither **-z** nor **-Z** is specified and the amount of free disk space becomes less than twice the total disk space, **savecrash** will compress the remaining files.

-D *dumpdevice*
dumpdevice is the name of the device containing the header of the raw crash image. The console messages from the time of the panic will identify the major and minor numbers of this device. This option, in combination with **-O**, can be used to tell **savecrash** where to find the dump in the rare instances that **savecrash** doesn't know where to look.

-O *offset*
offset is the offset in kBytes, relative to the beginning of the device specified with **-D** above, of the header of the raw crash image. The console messages from the time of the panic will identify this offset. This option, in combination with **-D**, can be used to tell **savecrash** where to find the dump in the rare instances that **savecrash** doesn't know where to look.

-d *sysfile*
sysfile is the name of a file containing the image of the system that produced the core dump (that is, the system running when the crash occurred). If this option is not specified, **savecrash** gets the file name from the dump itself. If the file containing the image of the system that caused the crash has changed, use this option to specify the new file name.

-m *minfree*
minfree is the amount of free space (in kBytes) that must remain free for ordinary user files after **savecrash** completes, in addition to space reserved for the superuser. If necessary, only part of the dump will be saved to achieve this requirement. *minfree* may be specified in bytes (**b**), kilobytes (**k**), megabytes (**m**), or gigabytes (**g**). The default *minfree* value is zero, and the default unit is kilobytes.

-s *chunksize*
chunksize is the size (default kBytes) of a single physical memory image file before compression. The kByte value must be a multiple of page size (divisible by 4) and between 64 and 1048576. *chunksize* may be specified in units of bytes (**b**), kilobytes (**k**), megabytes (**m**), or gigabytes (**g**). Larger numbers increase compression efficiency at the expense of both **savecrash** time and debugging time. If **-s** is not specified, a default is chosen based on the physical memory size and the amount of available file system space.

-t *tapedevice*
tapedevice is the tape device where the crash dump will be written. Crash dumps that are written to tape are written using a **tar** format. The crash dump tape can be read using *tar*(1).

When the **-t** option is specified, the **-p** option is not allowed and the whole dump is always preserved. In addition, **-c** and **-l**, are not allowed and **-m** is ignored. Also, when **-t** is specified, **savecrash** will not perform any compression.

When *dirname* is specified with the **-t** option, *dirname* is the name of the existing directory where the **INDEX** file is created; the default directory is **/tmp**. The **INDEX** file is the first file that is written out to the dump tape. This file is written a second time once all the dump files have been written. The first copy of the file only contains crash dump header information and its filename on tape is **tmpindex**. It does not contain information for the module and image files.

When writing to tape, the tape device must be online otherwise the command will fail with an error. Additionally, when **savecrash** reaches end-of-tape, it will prompt the user for the next tape. Any tape errors encountered will result in a generic tape error.

-w *opt* Defines the interaction between **savecrash** and **swapon**. *opt* can be one of the following values:

NOSWAP Do not run **swapon** from **savecrash**.

SWAPEACH (default) Call **swapon** each time **savecrash** finishes saving the image from each dump device. This option provides the most efficient use of paging space.

SWAPEND Only call **swapon** when **savecrash** finishes saving the image file from *all* dump devices. If this option is used, no additional paging space other than the primary paging space is available until the complete crash dump image is saved. This option provides a second chance to retrieve the crash image if **savecrash** fails on first attempt.

For compatibility with earlier **savecore** syntax, the values of **0**, **1** and **2** can be used in place of **NOSWAP**, **SWAPEACH**, and **SWAPEND**, respectively. This usage is obsolescent.

RETURN VALUE

Upon exit, **savecrash** returns the following values:

0	A crash dump was found and saved, *or* **savecrash** has preserved dump information from the primary swap device and is continuing to run in the background to complete its tasks.
1	A crash dump could not be saved due to an error.
2	No crash dump was found to save.
3	A partial crash dump was saved, but there was insufficient space to preserve the complete dump.
4	The *savecrash* process continued in the background, see the **INDEX** file for actual results.

WARNINGS

savecrash relies on the expectation that device numbers have the same meaning (point to the same devices) at the time the system dumps and at the time the dump is saved. If, after a crash, the system was booted from a different boot device in order to run **savecrash**, it is possible that this expectation will not be met. If so, **savecrash** may save an incomplete or incorrect dump or may fail to save a dump at all. Such cases cannot be reliably detected, so there may be no warning or error message.

If **savecrash** encounters an error while running in the background (such as running out of space), it will not be easily detectable by the caller. If the caller must ensure that the **savecrash** operation was successful, for example before writing to a dump device, the caller should specify **-f** to force **savecrash** to run in the foreground, and should then examine the exit status of the **savecrash** process when it finishes.

AUTHOR

savecrash was developed by HP and the University of California, Berkeley.

FILES

/etc/shutdownlog	shutdown log
/etc/rc.config.d/savecrash	savecrash startup configuration file
/sbin/init.d/savecrash	savecrash startup file
*dirname***/bounds**	crash dump number
/stand/vmunix	default kernel image saved by savecrash

SEE ALSO

adb(1), crashutil(1M), swapon(1M), tar(1).

NAME

scn - scan an HP SCSI disk array LUN for parity consistency

SYNOPSIS

scn -i *number_of_initiators device_file*

DESCRIPTION

scn scans the disks of the LUN in an HP SCSI disk array identified by the character device file *device_file*. The parity information for any block reporting inconsistent parity is corrected by an immediate call to **rpr**.

RETURN VALUE

scn returns the following values:

0	Successful completion.
-1	Command failed (an error occurred).

DIAGNOSTICS AND ERRORS

Errors can originate from problems with:

- **scn**
- SCSI (device level) communications
- system calls

Error messages generated by scn:

usage: scn -i <num initiators> <special>
scn encountered an error in command syntax. Enter the command again with all required arguments, in the order shown.

scn: LUN # too big
The LUN number, derived from the device file name, is out of range.

scn: Not a raw file
Utilities must be able to open the device file for raw access. That is, you must specify a character device file rather than a block device file.

scn: LUN does not exist
The addressed LUN is not configured, and is not known to the array controller.

scn: Not an HP SCSI disk array
The device being addressed is not an HP SCSI disk array.

SCSI (device level) communication errors:

Sense data associated with the failed operation is printed.

Error messages generated by system calls:

scn uses the following system calls:

malloc(), **free()**, **stat()**, **open()**, **close()**, **fopen()**, **fclose()**, **read()**, **write()**, and **ioctl()**.

Documentation for these HP-UX system calls contains information about the specific error conditions associated with each call. **scn** does not alter the value of **errno**. The interpretation of **errno** for printing purposes is performed by the system utility **strerror()**.

EXAMPLES

To scan the LUN at **/dev/rdsk/c2t0d0** on a Series 800 computer with two hosts (initiators) attached:

```
scn -i 2 /dev/rdsk/c2t0d0
```

DEPENDENCIES

The HP C2425 and HP C2427 disk arrays are only supported on Series 700 systems running HP-UX version 9.0X.

The HP C2430 disk array is supported on Series 700 and 800 systems running HP-UX versions 9.0X and 10.0X.

AUTHOR

scn was developed by HP.

SEE ALSO

rpr(1M).

NAME

scsictl - control a SCSI device

SYNOPSIS

scsictl [**-akq**] [**-c** *command*]... [**-m** *mode*[**=***value*]]... *device*

DESCRIPTION

The **scsictl** command provides a mechanism for controlling a SCSI device. It can be used to query mode parameters, set configurable mode parameters, and perform SCSI commands. The operations are performed in the same order as they appear on the command line.

device specifies the character special file to use.

Options

scsictl recognizes the following options.

-a Display the status of all mode parameters available, separated by semicolon-blank (**;**) or newline.

-c *command*
Cause the device to perform the specified command. *command* can be one of the following:

erase For magneto-optical devices that support write without erase, this command can be used to pre-erase the whole surface to increase data throughput on subsequent write operations. This command maintains exclusive access to the surface during the pre-erasure.

sync_cache
For devices that have an internal write cache, this command causes the device to flush its cache to the physical medium.

-k Continue processing arguments even after an error is detected. The default behavior is to exit immediately when an error is detected.

Command line syntax is always verified for correctness, regardless of the **-k** option. Improper command line syntax causes **scsictl** to exit without performing any operations on the device.

-m *mode* Display the status of the specified *mode* parameter. *mode* can be one of the following:

immediate_report
For devices that support immediate reporting, this mode controls how the device responds to write requests. If immediate report is enabled (**1**), write requests can be acknowledged before the data is physically transferred to the media. If immediate report is disabled (**0**), the device is forced to await the completion of any write request before reporting its status.

ir Equivalent to **immediate_report**.

queue_depth
For devices that support a queue depth greater than the system default, this mode controls how many I/Os the driver will attempt to queue to the device at any one time. Valid values are (**1**–**255**). Some disk devices will not support the maximum queue depth settable by this command. Setting the queue depth in software to a value larger than the disk can handle will result in I/Os being held off once a QUEUE FULL condition exists on the disk.

-m *mode***=***value*
Set the mode parameter *mode* to *value*. The available mode parameters and values are listed above.

Mode parameters that take only a binary value (**1** or **0**) can also be specified as either **on** or **off**, respectively.

-q Suppress the labels that are normally printed when mode parameters are displayed. Mode parameter values are printed in the same order as they appear on the command line, separated by semicolon-blank (**;**) or newline.

Mode parameters and commands need only be specified up to a unique prefix. When abbreviating a mode parameter or command, at least the first three characters must be supplied.

DIAGNOSTICS

Diagnostic messages are generally self-explanatory.

EXAMPLES

To display all the mode parameters, turn **immediate_report** on, and redisplay the value of **immediate_report**:

```
scsictl -a -m ir=1 -m ir /dev/rdsk/c0t6d0
```

producing the following output:

```
immediate_report = 0; queue_depth = 8; immediate_report = 1
```

The same operation with labels suppressed:

```
scsictl -aq -m ir=1 -m ir /dev/rdsk/c0t6d0
```

produces the following output:

```
0; 8; 1
```

WARNINGS

Not all devices support all mode parameters and commands listed above. Changing a mode parameter may have no effect on such a device.

Issuing a command that is not supported by a device can cause an error message to be generated.

scsictl is not supported on sequential-access devices using the tape driver.

The **immediate_report** mode applies to the entire device; the section number of the *device* argument is ignored.

To aid recovery, immediate reporting is not used for writes of file system data structures that are maintained by the operating system, writes to a hard disk (but not a magneto-optical device) through the character-device interface, or writes to regular files that the user has made synchronous with **O_SYNC** or **O_DSYNC** (see *open*(2) and *fcntl*(2)).

DEPENDENCIES

disc3

When the system is rebooted, the **disc3** driver always resets the value of the **immediate_report** mode parameter to **off**. If **ioctl()** or **scsictl** is used to change the setting of immediate reporting on a SCSI device, the new value becomes the default setting upon subsequent configuration (e.g., opens) of this device and retains its value across system or device powerfail recovery. However, on the next system reboot, the **immediate-report** mode parameter is again reset to the value of the tunable system parameter, **default_disk_ir**. This is set in the *system_file* used to create the HP-UX system by the **config** command (see *config*(1M)).

sdisk

If **ioctl()** or **scsictl** is used to change the setting of immediate reporting on a SCSI device, the new value becomes the default setting upon subsequent configuration (e.g., opens) of this device until the "last close" of the device, that is, when neither the system nor any application has the device open (for example, unmounting a file system via **umount** and then mounting it again via **mount** (see *mount*(1M)). On the next "first open", the **immediate-report** mode parameter is again reset to the value of the tunable system parameter, **default_disk_ir**. This is set in the *system_file* used to create the HP-UX system by the **config** command (see *config*(1M)).

SEE ALSO

config(1M), diskinfo(1M), fcntl(2), open(2).

NAME

see - access bytes in the HP SCSI disk array controller EEPROM

SYNOPSIS

see -d special

see -b *byte_number* **-h** *hex_byte device_file*

DESCRIPTION

see displays, or changes bytes in the controller EEPROM of the HP SCSI disk array associated with device file *device_file*. A 64-byte area in the EEPROM is accessible to the user. Although the command is directed to a single LUN, the EEPROM settings affect all the LUNs of the device.

Options

-d Display only. Displays the current values of the bytes in the accessible portion of the EEPROM.

-b *byte_number* **-v** *hex_byte*
Loads the hexadecimal value **hex_byte** into the decimal byte **byte_number** of the user accessible 64-byte region in the EEPROM.

BYTE DESCRIPTION

The following list of user accessible bytes in the EEPROM, and their default values is provided for informational purposes only. Changing the values can result in "incorrect" controller behavior with respect to HP SCSI disk array utilities, and other support software. See *WARNINGS*.

byte	meaning	C2425/7 value	C2430 value
0	enable synchronous negotiation	0x00	0x00
1	enable wide negotiation	0x00	0x00
2	spin-up algorithm	0x01	0x01
3	spin-up delay	0x32	0x1e
4	ready timeout	0x0a	0x17
5	host command delay at power on	0x00	0x00
6	firmware drive cmd timeout value	0x64	0x64
7	default RAID level	0x00	0x05
8	option control bits MSB	0x00	0x00
9	option control bits LSB	0x27	0x53
10	sense key for drive failures	0x06	0x06
11	inquiry data byte 7	0x12	0x32
12	ROM sequence control bits	0x01	0x01
13	synchronization control bits	0x02	0x02
14	inquiry revision level format	0x00	0x00
15	diagnostic self-test options	0x01	0x01
16	host command delay for bus reset	0x00	0x00
17	inquiry unconfigured device type	0x20	0x20
18	software command timeout value	0x14	0x14
19	software command timeout actions	0x07	0x07
20	drive bus reset to ready wait	0x08	0x08
21	host delay after data phase	0x00	0x00
22	drive scan disabled channel (MSB)	0x00	0x00
23	drive scan disabled channel (LSB)	0x00	0x00
24	time to asynchronous event	0x00	0x00
25	fan polling interval	0x00	0x00
26	power supply polling interval	0x00	0x00
27	reserved	0x00	0x00
28	Error Reporting Options (MSB)	0x01	0x01
29	Error Reporting Options (LSB)	0x00	0x00
30-63	reserved	0x00	0x00

RETURN VALUE

see returns the following values:

0 Successful completion.

-1 Command failed (an error occurred).

DIAGNOSTICS AND ERRORS

Errors can originate from problems with:

- **see**
- SCSI (device level) communications
- system calls

Error messages generated by see:

usage: see <-d | -b <byteno> -v <hex byte>> <special>
An error in command syntax has occurred. Re-enter command with the required arguments, in the order shown.

see: Arg out of range
One of the arguments has exceeded its maximum or minimum size, or is incorrect in form. Check the size and form of each argument.

see: device busy
To ensure that **see** does not modify a disk array that is being used by another process, **see** attempts to obtain exclusive access to the disk array. If the disk array is already opened by another process (for example, LVM — the Logical Volume Manager), a "**device busy**" error message is returned by the driver. To eliminate the "**device busy**" condition, determine what process has the device open. In the case of LVM, it is necessary to deactivate the volume group containing the array before configuring the array (see *vgchange*(1M)).

see: LUN # too big
The LUN number, which is derived from the device special file name, is out of range.

see: LUN does not exist
The addressed LUN is not configured, and thus is not known to the array controller.

see: Not a raw file
Utilities must be able to open the device file for raw access.

see: Not an HP SCSI disk array
The device being addressed is not an HP SCSI disk array.

see: Transfer length error
The amount of data actually sent to or received from the device was not the expected amount.

SCSI (device level) communication errors:

Sense data associated with the failed operation is printed.

Error messages generated by system calls:

see uses the following system calls:

stat(), **open()**, **close()**, **read()**, **write()**, and **ioctl()**.

Documentation for these HP-UX system calls contains information about the specific error conditions associated with each call. **see** does not alter the value of **errno**. The interpretation of **errno** for printing purposes is performed by the system utility **strerror()**.

EXAMPLES

To display the values of the accessible EEPROM bytes on HP SCSI disk array **/dev/rdsk/c2t6d0** on a Series 700:

```
see -d /dev/rdsk/c2t6d0
```

WARNING

Changing the values of EEPROM bytes can result in incorrect controller behavior with respect to utilities and support software that may not be immediately obvious. Also, the EEPROM can only be written to a finite number of times, and if its write count is exceeded, it must be replaced.

DEPENDENCIES

The HP C2425 and HP C2427 disk arrays are only supported on Series 700 systems running HP-UX version 9.0X.

The HP C2430 disk array is supported on Series 700 and 800 systems running HP-UX versions 9.0X and 10.0X.

AUTHOR

see was developed by HP.

NAME

sendmail - send mail over the Internet

SYNOPSIS

/usr/sbin/sendmail [*mode*] [*flags*] [*address* ...]

DESCRIPTION

sendmail sends a message to one or more *recipients* or *addresses*, routing the message over whatever networks are necessary. **sendmail** does internetwork forwarding as necessary to deliver the message to the correct place.

sendmail is not intended as a user interface routine. Other programs provide user-friendly front ends. **sendmail** is used only to deliver pre-formatted messages.

With no *flags* specified in the command line, **sendmail** reads its standard input up to an end-of-file or a line consisting only of a single dot (**.**) and sends a copy of the message found there to all of the addresses listed in the command line. It determines the network(s) to use based on the syntax and contents of the addresses, according to information in the **sendmail** configuration file. The default configuration file is **/etc/mail/sendmail.cf**.

Local addresses are looked up in a file and aliased appropriately, and **sendmail** also supports the use of NIS and LDAP for address lookup. Aliasing can be prevented by preceding the address with a backslash (****). Normally the sender is not included in any alias expansions. For example, if 'john' sends to 'group', and 'group' includes 'john' in the expansion, then the letter will not be delivered to 'john'.

If **newaliases** is invoked, **sendmail** will rebuild the alias database. **newaliases** is identical to **sendmail -bi**. See *newaliases*(1M). Mail that is temporarily undeliverable is saved in a mail queue. If **mailq** is invoked, **sendmail** will print the contents of the mail queue. The mail queue files are in the directory **/var/spool/mqueue**. **mailq** is identical to **sendmail -bp**. See *mailq*(1).

Arguments

sendmail recognizes the following arguments:

mode	A mode selected from those described in the "Modes" subsection below. Only one mode can be specified. The default is **-bm**.
address	The address of a recipient. Several addresses can be specified.
flags	A flag selected from those described in the "Flags" subsection below. Several flags can be specified.

Modes

sendmail operates in one of the following modes. The default is **-bm**, deliver mail in the usual way.

-ba Go into ARPANET mode. All input lines must end with a CR-LF, and all messages will be generated with a CR-LF at the end. Also, the "From:" and "Sender:" fields are examined for the name of the sender.

-bd Run as a daemon. **sendmail** will fork and run in background listening on socket 25 for incoming SMTP connections.

-bD Run as a daemon, but run in foreground.

-bh Print the persistent host status database.

-bH Purge the persistent host status database.

-bi Initialize the alias database for the mail aliases file. **newaliases** is identical to **sendmail -bi**. See *newaliases*(1M).

-bm Deliver mail in the usual way (default).

-bp Print a listing of the mail queue. **mailq** is identical to **sendmail -bp**. See *mailq*(1).

-bs Use the SMTP protocol as described in RFC821 on standard input and output. This flag implies all the operations of the *ba* flag that are compatible with SMTP.

-bt Run in address test mode. This mode reads addresses and shows the steps in parsing; it is used for debugging configuration tables.

-bv Verify names only - do not try to collect or deliver a message. Verify mode is normally used for validating users or mailing lists.

Flags

sendmail recognizes the following flags:

-B*type* Set the body type, 7BIT or 8BITMIME.

-C*file* Use alternate configuration file. **sendmail** refuses to run as root if an alternate configuration file is specified.

-d*X* Set debugging value to **X**.

-F*fullname* Set the full name of the sender.

-f*name* Sets the name of the "from" person (i.e., the sender of the mail) to **name**. If the user of the **-f** option is not a "trusted" user (normally *root*, *daemon*, and *network*) and if the **name** set using the **-f** option and the login name of the person actually sending the mail are not the same, it results in an **X-Authentication-Warning** in the mail header.

-h*N* Set the hop count to *N*. The hop count is incremented every time the mail is processed. When it reaches a limit, the mail is returned with an error message, the victim of an aliasing loop. If not specified, "Received:" lines in the message are counted.

-i Ignore dots alone on lines by themselves in incoming messages. This should be set if you are reading from a file.

-n Do not do aliasing.

-N*dsn* Set delivery status notification conditions. The valid conditions with which *dsn* can be set are as follows:

never	For no notifications.
failure	If delivery failed.
delay	If delivery is delayed.
success	When message is successfully delivered.

-O*option***=***value*
Set option *option* to a specified *value*. Options are described below in ""Processing Options."

-o*x***=***value* Set option *x* to the specified *value*. Options are described below in "Processing Options."

-p*protocol* Set the name of the protocol used to receive the message. This can be a simple protocol name such as "UUCP" or a protocol and hostname, such as "UUCP:ucbvax".

-q*time* Process saved messages in the queue at given intervals. If *time* is omitted, process the queue once. *time* is given as a tagged number, with **s** being seconds, **m** being minutes, **h** being hours, **d** being days, and **w** being weeks. For example, **-q1h30m** or **-q90m** would both set the timeout to one hour thirty minutes. If *time* is specified, **sendmail** will run in background. This option can be used safely with **bd**.

-qI*substr* Limit processed jobs to those containing *substr* as a substring of the queue id.

-qR*substr* Limit processed jobs to those containing *substr* as a substring of one of the recipients.

-qS*substr* Limit processed jobs to those containing *substr* as a substring of the sender.

-r*name* An alternate and obsolete form of the **f** flag.

-R*return* Set the amount of the message to be returned if the message bounces. The values that can be set for *return* are as follows:

full	To return the entire message
hdrs	To return only the headers.

-t Read message for recipients. To:, Cc:, and Bcc: lines will be scanned for recipient addresses. The Bcc: line will be deleted before transmission.

-U Initial (user) submission. This flag should always be set when **sendmail** is called from a user agent such as **mail** or **elm**. This flag should never be set when called from a network delivery agent such as **rmail**.

-v Go into verbose mode. Alias expansions will be announced, etc.

-V*envid* Set the original envelope identification. This is propagated across SMTP to servers that support DSN's (delivery status notification) and is returned in DSN-compliant error messages.

-X*logfile* Log all traffic in and out of mailers in the indicated log file. This should only be used as a last resort for debugging mailer bugs. It will log a lot of data very quickly.

-- Stop processing command flags and use the rest of the arguments as addresses.

Processing Options

There are also a number of processing options that may be set. Normally these will only be used by a system administrator. Options may be set either on the command line using the **-o** flag or in the configuration file, **/etc/mail/sendmail.cf**. The options are:

AliasFile=*file*
Use alternate alias file.

HoldExpensive
On mailers that are considered "expensive" to connect to, do not initiate immediate connection. This requires queuing.

CheckpointInterval=*N*
Checkpoint the queue file after every *N* successful deliveries (default 10). This avoids excessive duplicate deliveries when sending to long mailing lists interrupted by system crashes.

DeliveryMode=*x*
Set the delivery mode to *x*. Delivery modes are

i interactive (synchronous) delivery.

b background (asynchronous) delivery.

q queue only; expect the messages to be delivered the next time that the queue is run.

d deferred; the same as **q** except that database lookups (DNS and NIS lookups) are avoided.

ErrorMode=*x*
Set error processing to mode *x*. Valid modes are

m mail back the error message.

w "write" back the error message (or mail it back if the sender is not logged in).

p print the errors on the terminal (default).

q throw away error messages (only exit status is returned).

e do special processing for the BerkNet.

If the text of the message is not mailed back by modes **m** or **w** and if the sender is local to this machine, a copy of the message is appended to the file **dead.letter** in the sender's home directory.

SaveFromLine
Save UNIX -style From lines at the front of messages.

MaxHopCount=*N*
The maximum number of times a message is allowed to "hop" before it is considered in a loop.

IgnoreDots
Do not take dots on a line by themselves as a message terminator.

SendMimeErrors
Send error messages in MIME format.

ConnectionCacheTimeOut=*timeout*
Set connection cache timeout.

ConnectionCacheSize=*N*
Set connection cache size.

Loglevel=*n*
The log level.

MeToo Send to "me" (the sender) also if the sender is in an alias expansion.

CheckAliases
Validate the right hand side of aliases during a **newaliases** command. See *newaliases*(1M).

OldStyleheaders
If set, this message may have old style headers. If not set, this message is guaranteed to have new style headers (i.e., commas instead of spaces between addresses). If set, an adaptive algorithm is used that will correctly determine the header format in most cases.

QueueDirectory=*queuedir*
Select the directory in which to queue messages.

StatusFile=*file*
Save statistics in the named file.

Timeout.queuereturn=*time*
Set the timeout on undelivered messages in the queue to the specified time. After delivery has failed (e.g., because of a host being down) for this amount of time, the failed messages will be returned to the sender. The default is three days.

UserDatabaseSpec=*userdatabase*
If set, a user database is consulted to get forwarding information. You can consider this an adjunct to the aliasing mechanism, except that the database is intended to be distributed; aliases are local to a particular host.

ForkEachJob
Fork each job during queue runs. May be convenient on memory-poor machines.

SevenBitInput
Strip incoming messages to seven bits.

EightBitMode= *mode*
Set the handling of eight bit input to seven bit destinations. Mode can be set to the following values

m	Convert to seven-bit MIME format.
p	Pass it as eight bits.
s	Bounce the mail.

MInQueueAge= *timeout*
Sets how long a job must ferment in the queue in between attempts to send it.

DefaultCharSet=*charset*
Sets the default character set used to label 8-bit data that is not otherwise labeled.

DialDelay=*sleeptime*
If opening a connection fails, sleep for sleeptime seconds and try again. Useful on dial-on-demand sites.

NoRecipientAction=*action*
Set the behaviour when there are no recipient headers (To:, Cc: or Bcc:) in message to action. The different values that can be set for action are

none	Leaves the message unchanged.
add-to	Adds a **To:** header with the envelope recipients.
add-apparently-to	Adds an **Apparently-To:** header with the envelope recipients.

add-bcc Adds an empty **Bcc:**

add-to-undisclosed
Adds a header reading **To:undisclosed-recipients:**

MaxDaemonChildren=*N*
Sets the maximum number of children that an incoming SMTP daemon will allow to spawn at any time to *N*.

ConnectionRateThrottle=*N*
Sets the maximum number of connections per second to the SMTP port to *N*.

AutoRebuildAliases
If set, causes sendmail to rebuild the alias database when needed. Setting this option may cause excessive overhead and is not recommended.

DontProbeInterfaces
If set, this option is to turn off the inclusion of all the interface names in **$=w** on startup. In particular, if you have lots of virtual interfaces, this option will speed up startup. However, unless you make other arrangements, mail sent to those addresses will be bounced. This is useful for sending mail to hosts which have dynamically assigned names.

DontBlameSendmail=*options*
This *options* allows you to bypass some of **sendmail** file security checks at the expense of system security. This should only be used if you are absolutely sure you know the consequences. The available *options* for **DontBlameSendmail** are as follows:

Safe
AssumeSafeChown
ClassFileInUnsafeDirPath
ErrorHeaderInUnsafeDirPath
GroupWritableDirPathSafe
GroupWritableForwardFileSafe
GroupWritableIncludeFileSafe
GroupWritableAliasFile
HelpFileinUnsafeDirPath
WorldWritableAliasFile
ForwardFileInGroupWritableDirPath
IncludeFileInGroupWritableDirPath
ForwardFileInUnsafeDirPath
IncludeFileInUnsafeDirPath
ForwardFileInUnsafeDirPathSafe
IncludeFileInUnsafeDirPathSafe
MapInUnsafeDirPath
LinkedAliasFileInWritableDir
LinkedClassFileInWritableDir
LinkedForwardFileInWritableDir
LinkedIncludeFileInWritableDir
LinkedMapInWritableDir
LinkedServiceSwitchFileInWritableDir
FileDeliveryToHardLink
FileDeliveryToSymLink
WriteMapToHardLink
WriteMapToSymLink
WriteStatsToHardLink
WriteStatsToSymLink
RunProgramInUnsafeDirPath
RunWritableProgram

DontInitGroups=*True* | *False*
This can be set to true to prevent program deliveries from picking up extra group privileges.

MaxRecipientsPerMessage=*no_of_recipients*
This option limits the number of recipients, *no_of_recipients* that will be accepted in a single SMTP transaction. After this number is reached, **sendmail** starts returning

"452 Too many recipients" to all **RCPT** commands. This can be used to limit the number of recipients per envelope (in particular, to discourage use of the server for spamming). Note: a better approach is to restrict relaying entirely.

MaxHeadersLength=*max_header_length*
This option allows to specify a maximum length of the sum of all headers, *max_header_length* . This can be used to prevent a Denial-of-Service(DoS) attack.

RunAsUser= *user*
If set, causes **sendmail** to do a **setuid** to that *user* early in processing to avoid potential security problems. However, this means that **/var/spool/mqueue** directory owned by the *user* and all **.forward** and **:include:** files must be readable by that *user*, and all files to be written must be writable by that *user*, and all programs will be executed by that *user*. It is also incompatible with the **SafeFileEnvironment** option. In other words, it may not actually add much to security. However, it should be useful on firewalls and other places where users do not have accounts and the **aliases** file is well constrained.

SafeFileEnvironment=*option*
If set, files named as delivery targets must be regular files in addition to the regular checks. Also, if the *option* is non-null, then it is used as the name of a directory that is used as a *chroot*(2) environment for the delivery; the file names listed in an alias or forward should include the name of this root.

QueueSortOrder=*option*
This *option* can take two values (*host* or *priority* and *time*). Based on that, the queue will be sorted.

host This makes better use of the connection cache, but may delay more "interactive" messages behind large backlogs under some circumstances. This is a good option if you have high speed links or do not do lots of "batch" messages, but less good if you are using something like PPP on a 14.4 modem.

time This option causes the queue to be sorted strictly on the time of submission. This may cause a bad behaviour over slow lines and on nodes with heavy traffic. Also, this does not guarantee that jobs will be delivered in submission order unless you also set **DeliveryMode=queue**. In general, it should probably only be used on the command line, and only in conjunction with **-qRhost.domain**.

PrivacyOptions=*flag*
The available values for *flag* are

public	Allow open access.
needmailhelo	Insist on HELO (or EHLO) before the MAIL command.
needexpnhelo	Insist on HELO (or EHLO) before the EXPN command.
noexpn	Disallow EXPN command totally.
needvrfyhelo	Insist on HELO (or EHLO) before the VRFY command.
novrfy	Disallow VRFY command totally.
restrictmailq	Restrict mailq command.
restrictqrun	Restrict -q command-line flag.
noreceipts	Don't return success DSN's.
goaway	Disallow essentially all SMTP status queries.
authwarnings	Put **X-Authentication-Warning** headers in messages if HELO was not used inside SMTP transaction.
noverb	flag to disable the SMTP VERB command.
noetrn	flag to disable the SMTP ETRN command.

S

Aliases

You can set up system aliases and user forwarding. The **alias** and **.forward** files are described in the *aliases*(5) manpage.

EXIT STATUS

sendmail returns an exit status describing what it did. The codes are defined in <**sysexits.h**>:

EX_OK	Successful completion on all addresses.
EX_NOUSER	User name not recognized.
EX_UNAVAILABLE	Catchall meaning necessary resources were not available.
EX_SYNTAX	Syntax error in address.
EX_SOFTWARE	Internal software error, including bad arguments.
EX_OSERR	Temporary operating system error, such as "cannot fork" .
EX_NOHOST	Host name not recognized.
EX_TEMPFAIL	Message could not be sent immediately, but was queued.

AUTHOR

The **sendmail** command was developed by the University of California, Berkeley, and originally appeared in BSD 4.2. This version of HP-UX **sendmail** originally came from **sendmail 8.9.3**.

FILES

$HOME/.forward	User's mail forwarding file
$HOME/dead.letter	User's failed message file

Except for the **/etc/mail/sendmail.cf** file and the daemon process ID file, the below mentioned default pathnames are all specified in the configuration file, **/etc/mail/sendmail.cf**. These default file names can be overridden in the configuration file.

/etc/mail/aliases	raw data for alias names
/etc/mail/aliases.db	data base of alias names
/etc/mail/sendmail.cf	configuration file
/usr/share/lib/sendmail.hf	help file
/etc/mail/sendmail.st	collected statistics
/var/spool/mqueue/*	mail queue files
/etc/mail/sendmail.pid	The process id of the daemon
/etc/mail/sendmail.cw	The list of all hostnames that are recognized as local, which causes sendmail to accept mail for these hosts and attempt local delivery
/etc/nsswitch.conf	configuration file for the name-service switch

SEE ALSO

aliases(5), convert_awk(1M), elm(1), expand_alias(1), identd(1M), idlookup(1), killsm(1M), mail(1), mailq(1), mailstats(1), mailx(1), mtail(1M), newaliases(1M), praliases(1), smrsh(1M).

NAME

service.switch - indicate lookup sources and fallback mechanism

SYNOPSIS

`/etc/mail/service.switch`

DESCRIPTION

`/etc/mail/service.switch` is a *sendmail*(1M) service switch similar to **`/etc/nsswitch.conf`** (see *switch*(5)) that indicates the lookup source for hostnames and aliases. It consists of two lines, one for hosts and one for aliases. The lookup sources are listed after the 'hosts' or 'aliases' name. For hosts, one or more of the following can be listed: files (for **`/etc/hosts`**), dns, nis, or nisplus. For aliases, one or more of the following can be listed: files (for **`/etc/mail/aliases`**), nis, or nisplus.

Sample Configurations

1. The default configuration for service.switch is to use dns for hostname lookups and the aliases file for aliases. (Note that due to a bug, the hostname lookup will never fallback to a file lookup, so anything listed after dns will be ignored.)

```
hosts   dns files
aliases files
```

2. To work with a non-dns environment that uses file lookups (**`/etc/hosts`**), the following service.switch can be used:

```
hosts   files
aliases files
```

3. To work with a NIS environment that does not use DNS, the following service.switch can be used:

```
hosts   nis files
aliases nis files
```

4. To work with a NISPLUS environment that does not use DNS, the following service.switch can be used:

```
hosts   nisplus files
aliases nisplus files
```

Modifying SENDMAIL.CF

The **`sendmail.cf`** file must be modified to request the usage of the **`service.switch`** file. Otherwise, the default for **`sendmail.cf`** is to use DNS for host name lookups, and files for alias lookups. To use NIS, NISPLUS, or files, the following line must be uncommented in **`sendmail.cf`**:

```
#O ServiceSwitchFile=/etc/mail/service.switch
```

SEE ALSO

sendmail(1M).

HISTORY

The **`service.switch`** file appeared in sendmail V8.

S

NAME

set_parms - set up system hostname, networking, date/time and root password

SYNOPSIS

set_parms *hostname* | *timezone* | *date_time* | *root_passwd* | *ip_address* | *taddl_netwrk*

set_parms *initial*

DESCRIPTION

set_parms is an interactive system set up program which allows the user to set up various important system parameters when first booting up a newly installed operating system. In a first boot situation, **set_parms** is invoked automatically by **/sbin/auto_parms**. For **set_parms** purposes, first boot is defined as having no hostname set when the system starts up. This causes **set_parms** to step through all of its sub-areas to allow the user to set a hostname, select the proper timezone for the system's location, set the date and time, set a root password, and set an IP address, netmask, default routing, DNS, and NIS information.

After the system has booted and is running, **set_parms** may also be called directly from the command line to finish setting up a particular sub-area (first form above), or to step through all areas (via **set_parms** *initial*) similar to how it works at first boot. There are certain limitations to its actions when it's run after first boot, see below.

set_parms has two available user interfaces: a terminal-based interface and a graphical interface. At first boot, the graphical interface will be used if the system console is a graphics device, otherwise a terminal interface is used. If called on a normal running system, the graphical interface will be used if and only if a **DISPLAY** variable is set in the user's environment and an X server can be contacted at the host referenced in the **DISPLAY** variable. If run under CDE, **set_parms** should select the graphical interface.

set_parms is also DHCP-aware. If the user attempts to change DHCP supplied data such as the hostname or IP address, **set_parms** will issue a warning. If the user continues with the changes, then **set_parms** will relinquish the DHCP lease. On first boot, **set_parms** will ask the user if he would like to try getting set up data from a DHCP server.

set_parms can only be run by the super-user.

Options

Each sub-area is described below. In a first boot situation, all of the sub-areas are run sequentially. Special first boot behavior is noted if applicable, along with any concerns when calling **set_parms** on a running system. When calling a sub-area directly, only a unique portion of the sub-area name must be given.

hostname Sets the system hostname. Validates a user supplied hostname according to host naming conventions and sets various system initialization variables to operate with that hostname. Particularly, it edits **/etc/hosts** to associate the new hostname with the current IP address of the system, if that can be determined. *First boot :* Also allows the user to specify a DHCP server to get hostname and networking information from, then confirms the information.

WARNING: **set_parms** does not know about optionally installed software when changing the hostname. If such software remembers the previous hostname, then it may not work properly after the hostname is changed.

A mechanism is provided that helps generalize the hostname changing function. **set_parms** will call, in *ls*(1) sorted order, any executable programs installed in the directory **/sbin/ch_hostname.d**. This occurs in first boot or non-first boot calls. HP may in the future supply special programs in this location. The system administrator may also supply custom programs for site installations using, for example, *Ignite-UX*.

The system must be rebooted after the hostname is changed for it to take full effect.

timezone Allows the user to select a timezone based on the country of location. Also allows setting a user-supplied timezone. The system will need to be rebooted for a change to take effect.

date_time Allows the user to set the system date and time interactively. The change takes effect immediately.

root_passwd Allows the user to set or change the root password of the system. This function just calls the *passwd*(1) command.

ip_address	Allows the user to set or change the primary IP address of the system. An IP address change will require a system reboot to take effect. Edits the **/etc/hosts** file to associate the new IP address with the current hostname. *First boot:* also lets the user pick the lan interface to set up for this and subsequent networking functions. *Non-first boot:* In multiple lan systems, it assumes that the IP address is being changed for the *lowest numbered* IP address/Lan Interface Card data set in the **/etc/rc.config.d/netconf** file.
addl_netwrk	Allows the user to set the subnet mask, which defines the network and local portions of a network address, the default routing gateway, and define access to the Domain Name System (DNS) and Network Information Service (NIS). A reboot is required for everything to take effect in the non-first-boot case.

set_parms and Ignite-UX/Cold Install

After "cold installing" HP-UX from a CD, or using Ignite-UX to install HP-UX, the file **/tmp/install.vars** is generally left on the system. This file is used to communicate to **set_parms** hostname and networking information that was used during the installation, in case the user wants to use any of these parameters as final system parameters. In particular, **set_parms** uses as defaults the shell-style variables in this file that begin with **INST_**. For example, **INST_LAN_DEV** indicates which LAN interface was used during a network cold install. This is the LAN interface that **set_parms** will configure. In general, **set_parms** first looks in the system configuration files in the **/etc/rc.config.d** directory for default information, then in **/tmp/install.vars**.

If Ignite-UX is installed on your system, see the manual pages for *ignite*(5) and *instl_adm*(4). In particular, look at *instl_adm*(4) for descriptions of the *is_net_info_temporary*, *run_setparms*, and *final* variables.

Native Language Support (NLS):

set_parms supports all of the standard HP supplied languages. In first boot situations, the language **set_parms** uses will be dictated by either *geocustoms*(1M) or the LANG variable as set in **/etc/rc.config.d/LANG**. Geocustoms , if called by **set_parms**, allows the user to pick both a **system set up** language (the language of the system administrator setting up the system, to use in the user interface) and a **system default** language (the language of the end user). **set_parms** uses the **system set up** language as picked within geocustoms, *unless* the system default language is set to be an Asian language. In this case, it will use the Asian language, (from the file **/etc/rc.config.d/LANG**), since geocustoms itself does not provide an Asian language user interface.

Interaction with auto_parms and geocustoms

During the boot-up sequence, **/sbin/rc** always invokes **auto_parms**, which in turn detects the first boot situation and the need to run geocustoms (independent conditions) and calls **set_parms** if either or both of these conditions are true. **set_parms** first sets up an X-windows environment (if the system console is on a graphics display), and then calls geocustoms if necessary. After geocustoms (if called) is finished, and in a first boot situation, **set_parms** starts its interface and, based on user input, may call back into **auto_parms** to obtain and set up the management of a DHCP lease. After this has been done, and after **set_parms** completes its other system set-up actions, control passes back to **/sbin/rc** which completes the boot-up sequence utilizing the newly created system data.

EXAMPLES

See **/sbin/rc** for invocation context in the first boot case.

FILES

/sbin/set_parms
The main driver program.

/sbin/set_parms.util
Common subroutines used by **set_parms** and the sub-area programs.

/sbin/set_parms.d/
Directory which holds all of the sub-area programs called by **set_parms**. **set_parms** runs these in sorted order.

/sbin/ch_hostname.d/
Directory containing the hostname-change programs defined by the user. These are standalone programs run, in sorted order, by **set_parms** when setting or changing the hostname.

/tmp/install.vars
File set by Ignite-UX/Cold Install which contains networking and other system information used

during the installation.

SEE ALSO

auto_parms(1M), geocustoms(1M), dhcpdb2conf(1M), ignite(5), instl_adm(4).

NAME

setboot - display and modify boot variables in stable storage

SYNOPSIS

/usr/sbin/setboot [**-p** *primary-path*] [**-a** *alternate-path*] [**-b on**|**off**] [**-s on**|**off**]
[**-v**] [**-t** *testname*=**on**|**off**|**default**]... [**-T** *testname*=**on**|**off**|**default**]...

DESCRIPTION

The **setboot** command displays and sets boot variables in stable storage (also known as nonvolatile memory). Any user can display the values; only a superuser can change them.

On all systems, the variables are: primary path, alternate path, autoboot flag, and autosearch flag. If SpeedyBoot is installed, the variables expand to include: early CPU tests, late CPU tests, full memory tests, processor hardware tests, and central electronic complex tests.

With no options, **setboot** displays the current values for the primary and alternate boot paths and the autoboot and autosearch flags. If SpeedyBoot is installed, **setboot -v** also displays the status of the CPU, memory, hardware, and electronics tests.

SpeedyBoot

The SpeedyBoot firmware and software extensions allows a superuser to control which firmware tests are executed by the system during the boot process. The tests settings can be specified both for all subsequent boots and for the next one only. They are described in the "The Tests" section below.

The **-v**, **-t**, and **-T** options of the **setboot** command provide the user interface to the firmware tests.

SpeedyBoot augments the test control that is available on systems that have the Boot Console Handler (BCH). By turning off some or all of the boot tests, you can shorten boot time appreciably. However, in the event of a system panic or boot failure, all tests are executed on the subsequent boot.

Some older platforms can be upgraded to new firmware that supports SpeedyBoot.

SpeedyBoot Tests

The SpeedyBoot tests and the possible display values are summarized in the following table:

Test	Current	Supported	Default	NEXT BOOT
all	on|off|partial	yes|no|partial	on|off|partial	on|off|partial
SELFTESTS	on|off|partial	yes|no|partial	on|off|partial	on|off|partial
early_cpu	on|off	yes|no	on|off	on|off
late_cpu	on|off	yes|no	on|off	on|off
FASTBOOT	on|off|partial	yes|no|partial	on|off|partial	on|off|partial
full_memory	on|off	yes|no	on|off	on|off
PDH	on|off	yes|no	on|off	on|off
CEC	on|off	yes|no	on|off	on|off

The Columns

Test
The keyword names of the tests that can be controlled by SpeedyBoot. See "The Tests" section below.

Current
The current enablement of each test. **on** means the test is normally executed on each boot. **off** means the test is normally omitted on each boot. **partial** means some of the subtests are normally executed on each boot.

Supported
Whether the test is supported by the system firmware. **yes** means the test is supported. **no** means the test is not supported. **partial** means some of the subtests are supported.

Default
The default values for each test. **on**, **off**, and **partial** are the same as for **Current**.

NEXT BOOT
The values for each test that will be used on the next boot. If they are different from **Current**, the **Current** values will be reestablished after the next boot. **on**, **off**, and **partial** are the same as for **Current**.

The Tests

These are keywords for the hardware tests that are executed by processor-dependent code (PDC) or firmware upon a boot or reboot of the system.

`all` All the listed tests.

`SELFTESTS`
Includes the **`early_cpu`** and **`late_cpu`** tests. This is equivalent to the **`SELFTESTS`** option in the boot console handler (BCH) service menu. The difference is that **`setboot`** can control the subtests separately, while BCH cannot.

`early_cpu`
When on, run firmware, cache, and CPU-specific tests. Performed out of firmware. When off, skip the tests.

`late_cpu`
When on, run firmware, cache, and CPU-specific tests. Performed out of memory and therefore faster than the **`early_cpu`** tests. When off, skip the tests.

`FASTBOOT`
Includes the **`full_memory`** and **`PDH`** tests. This is equivalent to the **`FASTBOOT`** option in the boot console handler (BCH) service menu. The difference is that **`setboot`** can control the subtests separately, while BCH cannot.

Note: When **`FASTBOOT`** is on, the tests *are* performed, and vice versa.

`full_memory`
When on, run write/read-write/read tests on all memory locations. When off, only initialize memory.

`PDH` Processor-dependent hardware. When on, test a checksum of read-only memory (ROM). When off, do not.

`CEC` Central electronic complex. When on, test low-level bus converters and I/O chips. When off, do not. **`CEC`** is not available on all systems.

Options

The **`setboot`** command supports the following options:

(none)
Display the current values for the primary and alternate boot paths and the autoboot and autosearch flags. See example 2 in the "EXAMPLES: General" section.

`-p` *primary-path*
Set the primary boot path variable to *primary-path*. See *ioscan*(1M) for the correct format.

`-a` *alternate-path*
Set the alternate boot path variable to *alternate-path*. See *ioscan*(1M) for the correct format.

`-s on|off`
Enable or disable the autosearch sequence.

`-b on|off`
Enable or disable the autoboot sequence.

`-v` Display the current values for the primary and alternate boot paths and the autoboot and autosearch flags and a status table of the SpeedyBoot tests. See example 1 in the "EXAMPLES: SpeedyBoot" section.

`-t` *testname*=*value*
Change the value for the test *testname* in stable storage to *value* for all following boots. The changes are reflected in the **Current** and **NEXT BOOT** columns of the **`setboot -v`** display.

testname can be one of the following keywords, as described above in the "DESCRIPTION: SpeedyBoot Tests" section.

`all`
`SELFTESTS`
`early_cpu`
`late_cpu`
`FASTBOOT`

full_memory
PDH
CEC

value can be one of:

on	Enable the test.
off	Disable the test.
default	Reset the test to the system default, which is shown in the **Defaults** column of the **setboot -v** display.

-T *testname*=*value*
Change the values for the test *testname* for the next system boot only. The changes are reflected in the **NEXT BOOT** column of the **setboot -v** display. The change does not modify stable storage, so the permanent values, shown in the **Current** column, are restored after the boot.

testname and *value* are the same as for the **-t** option.

RETURN VALUE

The **setboot** command returns one of:

0	Successful completion
1	Failure

DIAGNOSTICS

The **setboot** command returns the following error messages:

"*bootpath***" is not a proper bootpath**

The boot path *bootpath* is not in correct format. See *ioscan*(1M) for the proper hardware path format.

cannot open /dev/kepd - *message*

setboot cannot open the kernel pseudo driver file **/dev/kepd**. The *message* explains why.

cannot set autoboot/autosearch flags

The autoboot or autosearch flag could not be set.

cannot set *type* **boot path**

setboot can't set the specified boot path. *type* may be **primary** or **alternate**.

error accessing boot path - *message*

The *message* explains why. For example, you may not have permission (not be superuser) to change parameters.

error accessing firmware - *message*

The firmware could not be read or written. The *message* explains why.

Invalid Arguments Passed to the invoked PDC routine

This is an internal error.

test not found in /etc/setboot_tests file

The test you specified is not defined for your system.

The firmware of your system does not support querying or changing SELFTEST and FASTBOOT settings except through the boot-time console interface, ie, BCH menu. Invoked PDC routine [**option**] **not implemented**

You have specified a SpeedyBoot option (**-t**, **-T**, or **-v**) and your system does not have the firmware to support it.

Unknown error *errornum* **encountered by setboot(1M)**

An unexpected error, number *errornum*, was encountered while **setboot** was processing Speedy-Boot parameters.

Warning: Autoboot flag must be on for autosearch flag to have effect

S

If the autoboot flag is off, automatic searching for a bootable system cannot occur, even if the autosearch flag is on.

warning: invalid data in /etc/setboot_tests

The **/etc/setboot_tests** file contains tests that are not supported by **setboot** on your system. Do not modify this file.

EXAMPLES

General

1. Set the primary path to **2/4.1.0** and enable the autoboot sequence:

```
setboot -p 2/4.1.0 -b on
```

2. Display the boot paths and auto flags:

```
setboot
```

displays

```
Primary bootpath : 2/0/1.6.0
Alternate bootpath : 2/0/2.0.0

Autoboot is ON (enabled)
Autosearch is ON (enabled)
```

SpeedyBoot

1. Display all current stable storage values.

```
setboot -v
```

displays

```
Primary bootpath : 10/0.0.0
Alternate bootpath : 10/12/5.0.0

Autoboot is ON (enabled)
Autosearch is OFF (disabled)

TEST             CURRENT     SUPPORTED     DEFAULT     NEXT BOOT
----             -------     ---------     -------     ---------
all              partial     partial       partial     partial
  SELFTESTS      partial     yes           on          partial
    early_cpu    off         yes           on          off
    late_cpu     on          yes           on          on
  FASTBOOT       partial     yes           on          partial
    full_memory  off         yes           on          off
    PDH          on          yes           on          on
  CEC            off         no            off         off
```

2. Enable **full_memory** and **PDH** tests and have those tests executed on all subsequent reboots.

```
setboot -t FASTBOOT=on
```

3. Disable the late processor tests and have those tests skipped on all subsequent reboots. If early CPU tests are **on** when this command is executed, the **SELFTESTS** state in BCH stays **on** while **setboot -v** shows it as **partial**.

```
setboot -t late_cpu=off
```

4. Reset all tests to the machine-shipped default values.

```
setboot -t all=default
```

5. Reset only the **FASTBOOT** (**full_memory** and **PDH**) tests to their default values.

```
setboot -t FASTBOOT=default
```

6. Cause the early and late CPU tests to be executed on the next system boot. The previously set test values take effect again after the single boot.

```
setboot -T SELFTESTS=on
```

7. Cause all tests to be skipped on the next reboot. The previously set test values will take effect for subsequent reboots.

```
setboot -T all=off
```

WARNINGS

The **setboot** command fails under the following circumstances:

- The number of writes to the stable storage exceeds the number allowed by the architecture implementation.
- Hardware failure.
- The implementation does not have memory for the alternate boot path, in which case, this variable is neither readable nor writable.

The interpretation of Autoboot and Autosearch has changed for systems that support hardware partitions. The firmware interprets the bits in combination and not as individual bits as done before. In order to provide the traditional behaviour of **setboot**, the user input for the autoboot and autosearch flags is internally mapped to the right combination to achieve the desired behaviour. As a side effect, when the user issues the **setboot** command without any options to get a display of the current settings, he will be displayed the new mapped values for autoboot and autosearch. The user may not be able to always see the values which he had initially set using **-b** or **-s** options.

To make full use of the new boot device configuration features available on the systems which support hardware partitions, please use the **pf** command from the boot console handler.

DEPENDENCIES

If SpeedyBoot is not installed on a system, options **-v**, **-t**, and **-T** will produce a diagnostic error.

AUTHOR

setboot was developed by HP.

FILES

/dev/kepd	Special device file used by the **setboot** command.
/etc/setboot_tests	Definitions of tests which can be viewed or controlled with the **-v**, **-t**, and **-T** options.

SEE ALSO

hpux(1M), ioscan(1M), isl(1M), mkboot(1M).

NAME

setext (vxfs) - set extent attributes

SYNOPSIS

/usr/sbin/setext [**-F vxfs**] [**-e** *extent_size*] [**-f** *flag*] [**-r** *reservation*] [**-V**] *file*

DESCRIPTION

setext specifies a fixed extent size for a file, and reserves space for a file. The file must already exist.

Options

-e *extent_size* Specify a fixed extent size. *extent_size* is the number of file system blocks to allocate for the extent. An *extent_size* of zero cancels previous fixed-size extents for the file and uses the default extent allocation policy. See *vxtunefs*(1M) for information on extent size parameters.

-F vxfs Specify the VxFS file system type.

-f *flags* The available allocation flags are

align
Specify that all extents must be aligned on *extent_size* boundaries relative to the start of allocation units.

chgsize
Immediately incorporate the reservation into the file and update the file's on-disk inode with size and block count information that is increased to include the reserved space. The space added to the file is not initialized. Only users with appropriate privileges can use the **chgsize** option.

contig
Specify that the reservation must be allocated contiguously.

noextend
Specify that the file may not be extended after the preallocated space is used.

noreserve
Specify that the reservation is not a persistent attribute of the file. Instead, the space is allocated until the final close of the file, when any space not used by the file is freed. The temporary reservation is not visible to the user (via *getext*(1M) or the **VX_GETEXT** ioctl, for example).

trim
Specify that the reservation is reduced to the current file size after the last close by all processes that have the file open.

-r *reservation* Preallocate space for *file*. *reservation* is specified in file system blocks.

-V Echo the completed command line, but do not execute the command. The command line is generated by incorporating the user-specified options and other information derived from **/etc/fstab**. This option allows the user to verify the command line.

S

NOTES

setext is available only with the HP OnLineJFS product.

You can specify multiple flags by entering multiple instances of **-f** on the command line.

You must specify the allocation flags with either the **-e** or **-r** option.

Only the **align** and **noextend** allocation flags are persistent attributes of the file and therefore visible via *getext*(1M) or the **VX_GETEXT** ioctl. Other allocation flags may have persistent effects, but are not visible as allocation flags.

In some cases, **fsadm** may reorganize the extent map of a file in such a way as to make it less contiguous. However, it will not change the geometry of a file that has a fixed extent size.

SEE ALSO

getext(1M), fsadm_vxfs(1M), vxtunefs(1M), vxfsio(7) (particularly the section on **VX_SETEXT**).

NAME

setmemwindow - changes the window id of a running program or starts a program in a particular memory window

SYNOPSIS

setmemwindow [**-cjnfbov**] [**-i** *WinId*] **-p** *pid* | *program* [*arg* ...]

DESCRIPTION

setmemwindow is the command that changes the window id of a running process or starts a specified *program* in a particular memory window.

If the **-p** option is specified with a non-zero *pid*, only the process' window id is changed, and any value specified for *program* is ignored.

The executable *program* is only executed if the process id *pid* is either **0** or unspecified.

Changing the window id for the running process does not mean the process immediately attaches to or creates objects using that window. The targeted process does not begin using the window until it exec's a new image.

setmemwindow, used as a wrapper for an existing executable, starts the *program* in the desired (see **-i** option below) memory window. In order to execute *program*, **setmemwindow** changes the window id, fork's a child and exec's *program* in the child process. The default behavior of **setmemwindow** is to wait until *program* finishes. If **-n** is specified, the waiting for *program* is bypassed and **setmemwindow** exits immediately after fork'ing the child.

If **-c** and **-j** are unspecified, the default behavior is to place the process into the window specified by *WinId*. If *WinId* exists, then the process is placed into that memory window. If no window exists with *WinId*, an unused window is allocated and associated with *WinId*. **-c** specifies the creation of a window. If the window already exists and **-c** is specified, the call fails. **-j** specifies the joining to an existing window. If the window does not exist and **-j** is specified, the call fails.

The **-f** option instructs the command to exec *program* regardless if the **setmemwindow** is unable to change the process' window to the specified *WinId*. The failure to create a specific window may be the effect of the lack of available memory windows - the underlying kernel has not been configured with enough memory windows (exit status ENOMEM) or the feature is not implemented (exit status ENOSYS).

Options

-i *WinId*
Specifies the desired memory window. *WinId* is a key to the desired memory window. *WinId* is a user specified value and should be one of the values contained in the file **/etc/services.window**. Applications extract the user key from **/etc/services.window** according to a unique string contained in **/etc/services.window**, using the **getmemwindow** command. (See *getmemwindow*(1M) and *services.window*(4).)

The kernel tries to locate an existing window with *WinId*. If one is found, that window is used. If no window with *WinId* is found, an unused entry is located and assigned to *WinId*.

The value **0** for *WinID* is special. If specified, the process/program is placed into the default global window instead of a unique window with id *WinId*.

If *WinID* is unspecified, the process and its children will run in a private memory window, and no other processes in the system can attach to this memory window. This memory window remains active until the process and its children terminate.

-p *pid* | *program* [*arg* ...]
Change the memory window for process *pid*, or start *program* in the specified memory window. If *program* has arguments (*arg* ...), they must also be specified.

If **-p** is unspecified or the value of *pid* is **0**, the calling process has its window id changed, and *program* is exec'ed.

If a non-zero process *pid* is specified, only the window in that process is changed, and *program* is ignored.

-c
Create a window with id *WinId* and attach the specified process to it. If *WinId* already exists the call fails.

S

-j Join an existing window with id *WinId* . The specified process attaches to an existing memory window. If no entry exists the call fails.

-n If *program* is exec'ed, the default behavior is to **waitpid** (see the *wait*(2) manual page) for the process to terminate. Specifying **-n** causes **setmemwindow** to exit after fork'ing the child (that will exec *program*).

-f The default behavior for **setmemwindow** is to exit without executing the user specified program if the memory window cannot be set. The failure to set the memory window may be caused by the lack of enough memory windows in the system, the memory windows feature is not implemented, a memory window with *WinId* could not be found in the attempt to join a memory window, or a memory window with the *WinId* was found in the attempt to create a memory window. The **-f** option instructs **setmemwindow** to exec *program* regardless if the desired memory window was set or not. Obviously, using this option there is no guarantee *program* has been attached to the desired memory window and it is unclear in what memory window it is running. Using this option is strongly discouraged.

-b Create a memory window where both memory window quadrants use the same space id. For **SHMEM_MAGIC** executables this generates two quadrants with the same space id. Applications can use this to generate the appearance of larger contiguous shared memory ranges with a maximum of 2 gigabytes. For example, an application that generates a 1 gigabyte shared memory segment has that segment placed into the 2nd quadrant by default. If the application creates another 1 gigabyte segment that segment is placed in the 3rd quadrant. Both segments are contiguous virtually, allowing the application to treat the virtual range as if it were a contiguous 2 gigabyte segment.

This option only benefits **SHMEM_MAGIC** executables. They are the only type of executable format able to place shared objects in the 2nd quadrant.

-o Send the pid of the exec'ed *program* to the stdout. The message sent out is: "setmemwindow:pid=dddd\n", where dddd is the decimal value of the pid.

Application Usage

Memory Windows helps alleviate the 1.75 gigabytes limitation on system wide shared memory for 32-bit applications by allowing cooperating applications to configure their own 1 gigabyte window of shared resources.

The definition of a memory window is only available for 32-bit processes.

Note that memory windows allows the creation of more than 1.75 gigabytes of *total system wide* shared memory, but it does not extend how much shared memory a *single* process can create. **SHARED_MAGIC** executables are still limited to 1.75 gigabytes.

HP-UX ships memory windows disabled. To enable memory windows, the kernel tunable parameter, **max_mem_window**, must be set to the desired number. **max_mem_window** represents the number of memory windows beyond the global default window. Setting **max_mem_window** to **2**, for example, would produce a total of three memory windows, the default global window plus two user defined windows. Setting **max_mem_window** to **0** leaves only the default or global memory window.

There are two new commands and one file introduced by memory windows: **setmemwindow**, **getmemwindow**, and **/etc/services.window** file.

The **/etc/services.window** file maps a memory window application to a particular window id. Using this central file allows applications to share memory windows, by using the same window id, as well as avoid unintentional memory window collisions. See *services.window*(4) for more information.

The **getmemwindow** command is used to extract the window id of a user process from the **/etc/services.window** file. The **setmemwindow** command starts a particular process in a memory window. A common usage of these commands is to extract a memory window id with **getmemwindow**, which is then passed to **setmemwindow** to start a process with the given window id.

Processes must be in the same window to share data. Processes wanting to share global data, such as shared memory or **MAP_SHARED** memory mapped files, must make sure all processes are in the same memory window. If processes in different memory windows wish to share data reliably, the creator of the data must take steps to guarantee the data is placed in a location accessible to all processes.

For more detailed information on memory windows, refer to the *11.0 Memory Window White Paper*.

RETURN VALUE

The returned exit value is 0 on success or a positive number on failure.

If **-n** is not specified, the value returned is the exit status of the executed *program* obtained from the *wait-pid*(2) system call.

EXAMPLES

```
#
# Start the program "myprog" in a memory window extracted by the string
# "myapp".
#
WinId=$(getmemwindow myapp)
setmemwindow -i $WinId myprog arg1 arg2

#
# Start the program "myprog" in a newly created memory window
# extracted by the string "myapp".
#
WinId=$(getmemwindow myapp)
setmemwindow -c -i $WinId myprog arg1 arg2

#
# Start the program "myprog" in an existing memory window
# extracted by the string "myapp".
#
WinId=$(getmemwindow myapp)
setmemwindow -j -i $WinId myprog arg1 arg2

#
# Start the program "myprog" in a private memory window.  Only
# "myprog" and its descendents can access the window.
#
setmemwindow myprog arg1 arg2
```

AUTHOR

setmemwindow was developed by HP.

FILES

/etc/services.window File containing applications' associated window ids.

SEE ALSO

getmemwindow(1M), services.window(4), *11.0 Memory Windows White Paper*.

NAME

setmnt - establish the file-system mount table, **/etc/mnttab**

SYNOPSIS

/usr/sbin/setmnt

DESCRIPTION

The **setmnt** command creates the **/etc/mnttab** table (see *mnttab*(4)), which is needed by both the **mount** and **umount** commands (see *mount*(1M)). **setmnt** reads the standard input and creates an entry in **/etc/mnttab** for each line of input. Input lines have the format:

filesys node

where *filesys* is the name of the device special file associated with the file system (such as **/dev/dsk/c0t5d0**) and *node* is the root name of that file system. Thus *filesys* and *node* become the first two strings in the mount table entry.

WARNINGS

The **mount** and **umount** commands rewrite the **/etc/mnttab** file whenever a file system is mounted or unmounted if **/etc/mnttab** is found to be out of date with the mounted file system table maintained internally by the HP-UX kernel. The **syncer** command also updates **/etc/mnttab** if it is out of date (see *syncer*(1M)).

/etc/mnttab should never be manually edited. Use of this command to write invalid information into **/etc/mnttab** is strongly discouraged.

The **setmnt** command is not intented to be run interactively; input should be directed to it from a file (for example, **setmnt < /tmp/file.mnt**). If run interactively, terminate input with a **ctrl-D**.

setmnt silently enforces an upper limit on the maximum number of **/etc/mnttab** entries.

It is unwise to use **setmnt** to create false entries for **mount** and **umount**.

This command is obsolete and it may not be available for future releases.

FILES

/etc/mnttab	Mounted file system table

SEE ALSO

devnm(1M), mount(1M), syncer(1M), mnttab(4).

STANDARDS CONFORMANCE

setmnt: SVID2, SVID3

NAME

setprivgrp - set special privileges for groups

SYNOPSIS

setprivgrp *groupname* [*privileges*]

setprivgrp -g [*privileges*]

setprivgrp -n [*privileges*]

setprivgrp -f *file*

DESCRIPTION

The **setprivgrp** command associates a group with a list of privileges, thus providing access to certain system capabilities for members of a particular group or groups. The privileges can be displayed with the **getprivgrp** command (see *getprivgrp*(1)).

Privileges can be granted to individual groups, as defined in the **/etc/group** file, and globally for all groups.

Only a superuser can use the **setprivgrp** command.

Options and Arguments

setprivgrp recognizes the following options and arguments:

privileges	One or more of the keywords described below in "Privileged Capabilities".
groupname	The name of a group defined in the file named **/etc/group**. The current privileges for *groupname*, if any, are replaced by the specified *privileges*. To retain prior privileges, they must be respecified.
-g	Specify global privileges that apply to all groups. The current privileges, if any, are replaced by the specified *privileges*, To retain prior privileges, they must be respecified.
-n	If no *privileges* are specified, delete all privileges for all groups, including global privileges. If one or more *privileges* are specified, delete the specified privileges from the current privilege lists of all groups, including the global privilege list, but do not delete unspecified privileges.
-f *file*	Set the privileges according to entries in the file *file*. This file is usually **/etc/privgroup**. The entry formats are described below in "Group Privileges File Format".

Privileged Capabilities

The following system capabilities can be granted to groups:

CHOWN	Can use **chown()** to change file ownerships (see *chown*(2)).
LOCKRDONLY	Can use **lockf()** to set locks on files that are open for reading only (see *lockf*(2)).
MLOCK	Can use **plock()** to lock process text and data into memory, and the **shmctl()** **SHM_LOCK** function to lock shared memory segments (see *plock*(2) and *shmctl*(2)).
RTPRIO	Can use **rtprio()** to set real-time priorities (see *rtprio*(2)).
RTSCHED	Can use **sched_setparam()** and **sched_setscheduler()** to set POSIX.4 real-time priorities (see *rtsched*(2)).
SERIALIZE	Can use **serialize()** to force the target process to run serially with other processes that are also marked by this system call (see *serialize*(2)).
SETRUGID	Can use **setuid()** and **setgid()** to change, respectively, the real user ID and real group ID of a process (see *setuid*(2) and *setgid*(2)).

Group Privileges File Format

The file specified with the **-f** option should contain one or more lines in the following formats:

groupname [*privileges*]

-g [*privileges*]

-n [*privileges*]

They are described above in "Options and Arguments".

RETURN VALUE

setprivgrp exits with one of the following values:

0	Successful completion.
>0	Failure.

AUTHOR

setprivgrp was developed by HP.

FILES

/etc/group
/etc/privgroup

SEE ALSO

getprivgrp(1), chown(2), getprivgrp(2), lockf(2), plock(2), rtprio(2), rtsched(2), serialize(2), setgid(2), setuid(2), shmctl(2), privgrp(4).

NAME

setuname - change machine information

SYNOPSIS

setuname [**-s** *name*] [**-n** *node*] [**-t**]

DESCRIPTION

The **setuname** command is used to modify the value for system name and/or the node name by using the appropriate option(s).

The **setuname** command attempts to change the parameter values in both the running kernel and the system configuration to cross reboots. A temporary change affects only the running kernel.

Options

The **setuname** command supports the following options:

-s *name*	Changes the system name (e.g., HP-UX) in the **sysname** field of the **utsname** structure where *name* is the new system name and consists of alphanumeric characters and the special characters dash, underbar, and dollar sign.
-n *node*	Changes the name in the **nodename** field of the **utsname** structure where *node* specifies the new node name and consists of alphanumeric characters and the special characters dash, underbar, and dollar sign.
-t	Signifies a temporary change. The change will not survive a reboot.

Either or both of the **-s** or **-n** options must be given when invoking **setuname**.

The size of the *name* and *node* is limited to **UTSLEN**−1 characters. **UTSLEN** is defined in **<sys/utsname.h>**. Only users having appropriate privileges can use this command.

EXAMPLES

To permanently change the system name to **HP-UX** and the node name to **the-node**, issue the following command:

```
setuname -s HP-UX -n the-node
```

To temporarily change the system name to **SYSTEM** and the node name to **new-node**, issue the following command:

```
setuname -s SYSTEM -n new-node -t
```

SEE ALSO

uname(1), uname(2).

NAME

showmount - show all remote mounts

SYNOPSIS

/usr/sbin/showmount [**-a**] [**-d**] [**-e**] [*host*]

DESCRIPTION

showmount lists all clients that have remotely mounted a filesystem from *host*. This information is maintained by the **mountd** server on *host* (see *mountd*(1M)). The default value for *host* is the value returned by **hostname** (see *hostname*(1)).

Options

-a Print all remote mounts in the format

name:*directory*

where *hostname* is the name of the client, and *directory* is the directory or root of the file system that was mounted.

-d List directories that have been remotely mounted by clients.

-e Print the list of exported file systems.

WARNINGS

If a client crashes, executing **showmount** on the server will show that the client still has a file system mounted. In other words, the client's entry is not removed from **/etc/rmtab** until the client reboots and executes:

```
umount -a
```

Also, if a client mounts the same remote directory twice, only one entry appears in **/etc/rmtab**. Doing a **umount** of one of these directories removes the single entry and **showmount** no longer indicates that the remote directory is mounted.

AUTHOR

showmount was developed by Sun Microsystems, Inc.

SEE ALSO

hostname(1), exportfs(1M), mountd(1M), exports(4), rmtab(4).

S

NAME

shutdown - terminate all processing

SYNOPSIS

/sbin/shutdown [**-h**|**-r**] [**-y**] [**-o**] [*grace*]

/sbin/shutdown -R [**-H**] [**-y**] [**-o**] [*grace*]

DESCRIPTION

The **shutdown** command is part of the HP-UX system operation procedures. Its primary function is to terminate all currently running processes in an orderly and cautious manner. **shutdown** can be used to put the system in single-user mode for administrative purposes such as backup or file system consistency checks (see *fsck*(1M)), to halt or reboot the system, or to make the partition ready for reconfig. By default, **shutdown** is an interactive program.

Options and Arguments

shutdown recognizes the following options and arguments.

-h	Shut down the system and halt.
-r	Shut down the system and reboot automatically.
-R	Shut down the system to a ready to reconfig state and reboot automatically. This option is available only on systems that support hardware partitions.
-H	Shut down the system to a ready to reconfig state and do not reboot. This option can be used only in combination with the **-R** option. This option is available only on systems that support hardware partitions.
-y	Do not require any interactive responses from the user. (Respond **yes** or **no** as appropriate to all questions, such that the user does not interact with the shutdown process.)
-o	When executed on the cluster **server** in a diskless cluster environment, shutdown the **server** only and do not reboot **clients**. If this argument is not entered the default behavior is to reboot all **clients** when the **server** is shutdown.
grace	Either a decimal integer that specifies the duration in seconds of a grace period for users to log off before the system shuts down, or the word **now**. The default is 60. If *grace* is either 0 or **now**, **shutdown** runs more quickly, giving users very little time to log out.

If **-r** (reboot) or **-h** (halt) or **-R** (reconfig) are not specified, **standalone** and **server** systems are placed in single-user state. Either **-r** (reboot) or **-h** (halt) must be specified for a **client**; shutdown to single-user state is not allowed for a **client**. See *dcnodes*(1M), *init*(1M).

Shutdown Procedure

shutdown goes through the following steps:

- The **PATH** environment variable is reset to **/usr/bin:/usr/sbin:/sbin**.
- The **IFS** environment variable is reset to space, tab, newline.
- The user is checked for authorization to execute the **shutdown** command. Only authorized users can execute the **shutdown** command. See FILES for more information on the **/etc/shutdown.allow** authorization file.
- The current working directory is changed to the root directory (**/**).
- All file systems' super blocks are updated; see *sync*(1M). This must be done before rebooting the system to ensure file system integrity.
- The real user ID is set to that of the superuser.
- A broadcast message is sent to all users currently logged in on the system telling them to log out. The administrator can specify a message at this time; otherwise, a standard warning message is displayed.
- The next step depends on whether a system is **standalone**, a **server**, or a **client**.
  - If the system is **standalone**, **/sbin/rc** is executed to shut down subsystems, unmount file systems, and perform other tasks to bring the system to run level 0.

  - If the system is a **server**, the optional **-o** argument is used to determine if all **clients** in the cluster should also be rebooted. The default behavior (command line parameter **-o** is not entered) is to reboot all **clients** using **/sbin/reboot;** entering **-o** results in the **server** only being rebooted and the **clients** being left alone. Then **/sbin/rc** is executed to shut down subsystems, unmount file systems, and perform other tasks to bring the system to run level 0.
  - If the system is a **client**, **/sbin/rc** is executed to bring the system down to run-level 2, and then **/sbin/reboot** is executed. Shutdown to the single-user state is not an allowed option for **clients.**

- The system is rebooted, halted, or put in the ready to reconfig state by executing **/sbin/reboot** if the **-h** or **-r** or **-R** option was chosen. If the system was not a cluster client and the system was being brought down to single-user state, a signal is sent to the **init** process to change states (see *init*(1M)).

DIAGNOSTICS

device busy

This is the most commonly encountered error diagnostic, and happens when a particular file system could not be unmounted; see *mount*(1M).

user not allowed to shut down this system

User is not authorized to shut down the system. User and system must both be included in the authorization file **/etc/shutdown.allow**.

EXAMPLES

Immediately reboot the system and run HP-UX again:

```
shutdown -r 0
```

Halt the system in 5 minutes (300 seconds) with no interactive questions and answers:

```
shutdown -h -y 300
```

Go to run-level **s** in 10 minutes:

```
shutdown 600
```

Immediately shut down a partition so that it can be deleted:

```
shutdown -R -H 0
```

Reboot a partition in 5 minutes so that new cells that have been assigned to the partition become active:

```
shutdown -R 300
```

WARNINGS

The user name compared with the entry in the **shutdown.allow** file is obtained using **getlogin()** or, if that fails, using **getpwuid()** (see *getlogin*(3C) and *getpwent*(3C)).

The hostname in **/etc/shutdown.allow** is compared with the hostname obtained using **gethost-byname()** (see *gethostent*(3N)).

shutdown must be executed from a directory on the root volume, such as the **/** directory.

The maximum broadcast message that can be sent is approximately 970 characters.

When executing **shutdown** on an NFS diskless cluster server and the **-o** option is not entered, clients of the server will be rebooted. No clients should be individually rebooted or shutdown while the cluster is being shutdown.

FILES

/etc/shutdown.allow

Authorization file.

The file contains lines that consist of a system host name and the login name of a user who is authorized to reboot or halt the system. A superuser's login name must be included in this file in order to execute **shutdown**. However, if the file is missing or of zero length, the **root** user can run the **shutdown** program to bring the system down.

This file does not affect authorization to bring the system down to single-user state for maintenance purposes; that operation is permitted only when invoked by a superuser.

A comment character, **#**, at the beginning of a line causes the rest of the line to be ignored (comments cannot span multiple lines without additional comment characters). Blank lines are also ignored.

The wildcard character **+** can be used in place of a host name or a user name to specify all hosts or all users, respectively (see *hosts.equiv*(4)).

For example:

```
# user1 can shut down systemA and systemB
systemA user1
systemB user1
# root can shut down any system
+ root
# Any user can shut down systemC
systemC  +
```

SEE ALSO

dcnodes(1M), fsck(1M), init(1M), killall(1M), mount(1M), reboot(1M), sync(1M), dcnodes(3X), gethostent(3N), getpwent(3C), hosts.equiv(4).

For more information about shutdowns and reboots on Superdome systems, see the manual, *Managing Superdome Complexes: A Guide for System Administrators* available on the web at **docs.hp.com**.

NAME

sig_named - send signals to the domain name server

SYNOPSIS

sig_named [**-v**] [**debug** [**+**] *debug-level* | **dump** | **kill** | **restart** | **stats** | **trace**]

DESCRIPTION

sig_named sends the appropriate signal to the domain name server **/usr/sbin/named**. The process ID is obtained from **/var/run/named.pid** or from *ps*(1) if **/var/run/named.pid** does not exist.

Options

sig_named recognizes the following options and command-line arguments:

-v Verify that the name server is running before sending the signal. The verification is done using **ps** (see *ps*(1)).

debug [**+**]*debug-level*
Set the debugging output sent to **/var/tmp/named.run** to *debug-level*. If debugging is already on, it is turned off before the debug level is set. If **+** precedes *debug-level*, the current debugging level is raised by the amount indicated. If *debug-level* is zero, debugging is turned off.

dump Signal the name server to dump its database. The database is dumped to **/var/tmp/named_dump.db**.

kill Kill the name server process.

restart Signal the name server to reload its database.

stats Remove the old statistics file, **/var/tmp/named.stats**. Signal the name server to dump its statistics. Show the statistics file on the standard output.

trace Toggles tracing of incoming queries to **/var/adm/syslog/syslog.log.**

AUTHOR

sig_named was developed by HP.

FILES

/var/run/named.pid	Process ID
/var/tmp/named.run	Debug output
/var/tmp/named_dump.db	Dump of the name server database
/var/tmp/named.stats	Nameserver statistics data

SEE ALSO

kill(1), named(1M).

NAME

smrsh - restricted shell for sendmail

SYNOPSIS

smrsh -c *command*

DESCRIPTION

The **smrsh** program is intended as a replacement for **sh** for use in the **prog** mailer in **sendmail** configuration files. It sharply limits the commands that can be run using the **|program** syntax of **sendmail** in order to improve the overall security of your system. Briefly, even if a "bad guy" can get **sendmail** to run a program without going through an alias or forward file, **smrsh** limits the set of programs that he or she can execute.

Briefly, **smrsh** limits programs to be in the directory **/var/adm/sm.bin**, allowing the system administrator to choose the set of acceptable commands. It also rejects any commands with the characters \, **<**, **>**, **|**, **;**, **&**, **$**, **(**, **)**, **\r** (carriage return), and **\n** (newline) on the command line to prevent "end run" attacks.

Initial pathnames on programs are stripped, so forwarding to **/usr/ucb/vacation**, **/usr/bin/vacation**, **/home/server/mydir/bin/vacation**, and **vacation** all actually forward to **/var/adm/sm.bin/vacation**.

System administrators should be conservative about populating **/var/adm/sm.bin**. Reasonable additions are **vacation** and **rmail**. Do not include any shell or shell-like program (such as **perl**) in the **sm.bin** directory. Note that this does not restrict the use of shell or perl scripts in the **sm.bin** directory (using the **#!** syntax); it simply disallows execution of arbitrary programs.

FILES

/var/adm/sm.bin	Directory for restricted programs

SEE ALSO

sendmail(1M).

S

NAME

snmpd, snmpdm - Simple Network Management Protocol (SNMP) Daemon

SYNOPSIS

/usr/sbin/snmpd [**-a**] [**-authfail**] [**-C** *contact*] [**-Contact** *contact*] [**-h**] [**-help**] [**-L** *location*] [**-Location** *location*] [**-l** *logfile*] [**-logfile** *logfile*] [**-m** *logmask*] [**-mask** *logmask*] [**-n**] [**-P** *portnum*] [**-Port** *portnum*] [**-sys** *description*] [**-sysDescr** *description*]

/usr/sbin/snmpd [**-e** *extendFile*]

/usr/sbin/snmpdm [**-a**] [**-authfail**] [**-C** *contact*] [**-Contact** *contact*] [**-h**] [**-help**] [**-L** *location*] [**-Location** *location*] [**-l** *logfile*] [**-logfile** *logfile*] [**-m** *logmask*] [**-mask** *logmask*] [**-n**] [**-P** *portnum*] [**-Port** *portnum*] [**-sys** *description*] [**-sysDescr** *description*]

DESCRIPTION

The Master SNMP Agent (**/usr/sbin/snmpdm**) and the collection of subAgents (**/usr/sbin/mib2agt**, **/usr/sbin/hp_unixagt**, ...) that would attach to the Master Agent collectively form a single SNMP Agent. The SNMP Agent accepts SNMP Get, GetNext and Set requests from an SNMP Manager which cause it to read or write the Management Information Base (**MIB**). The **MIB** objects are instrumented by the subAgents.

The Master Agent can bind to three kinds of subAgents, namely,

- Loosely coupled subAgents or separate process subAgents which open IPC communication channels to communicate with the Master Agent,
- Shared library subAgents which are dynamically linkable libraries,
- Remotely coupled subagents which could run on a different processor or operating system and communicate with the Master Agent using TCP.

Options

The Master agent **/usr/sbin/snmpdm** and the script **/usr/sbin/snmpd** recognize the following command line options:

-authfail
-a Suppress sending authenticationFailure traps.

-Contact *contact*
-C *contact* Specify the contact person responsible for the network management agent. This option overrides the contact person specified in the Master Agent configuration file **/etc/SnmpAgent.d/snmpd.conf**. It does not alter the value specified in the file. By default, the agent's contact is a blank string. To configure the agent's contact, add the contact after the word contact: in the configuration file **/etc/SnmpAgent.d/snmpd.conf** or use the **-C** option.

-e *extendFile* This option is provided for backward compatibility with the pre-emanate snmpd.ea extensible SNMP agent. It is applicable only to the script **/usr/sbin/snmpd**, and only if the EMANATE extensible agent is installed. It is installed if the file **/usr/sbin/extsubagt** exists. This option causes the extsubagt to use the command line specified extendFile instead of the default file **/etc/SnmpAgent.d/snmpd.extend** to add user defined MIB objects to the SNMP agent.

-help
-h Display command line options and log mask values.

-Location *location*
-L *location* Specify the location of the agent. This option overrides the location specified in **/etc/SnmpAgent.d/snmpd.conf**. It does not alter the value in **/etc/SnmpAgent.d/snmpd.conf**. By default, the agent's location is a blank string. To configure the agent's location, add the location to **/etc/SnmpAgent.d/snmpd.conf** or use the **-L** option.

-logfile *logfile*
-l *logfile* Use the *logfile* for logging rather than the default logfile, **/var/adm/snmpd.log**. A value of **-** will direct logging to **stdout**.

-mask *logmask*
-m *logmask* Sets the initial logging mask to *logmask*. The logmask option may not be used in the agent start up scripts. This option should be used only while debugging the agent. See the **SNMP Agent Logging** section for valid values of logmask and for other details.

-n Normally **snmpdm** puts itself into the background as if the command was terminated with an ampersand(&). This option inhibits that behavior and makes the agent run in the foreground.

-Port *portnum*
-P *portnum* Specify the UDP port number that the agent will listen on for SNMP requests. The default is port 161. The value can also be specified in **/etc/services**. Only the super-user can start **snmpdm** and only one **snmpdm** can execute on a particular UDP port.

-sysDescr *description*
-sys *description*
Allows the user to specify the value for the **system.sysDescr MIB** object. The format is a text string enclosed in quotes. This option overrides the **sysDescr** specified in **/etc/SnmpAgent.d/snmpd.conf**. For example,

```
snmpdm -sys "nsmd1, test system"
```

SNMPv1 Security

An **SNMP** Manager application can request to read a **MIB** value available at an agent by issuing an **SNMP** GetRequest, or a manager application may request to alter a **MIB** value by issuing an **SNMP** SetRequest. Each **SNMP** request is accompanied by a community name, which is essentially a password that enables **SNMP** access to **MIB** values on an agent.

Note, the agent does not respond to any **SNMP** requests, including GetRequests, if no community name is configured in **/etc/SnmpAgent.d/snmpd.conf**. To configure the agent to respond to GetRequests accompanied by a specific community name, add the community name as a **get-community-name** to the configuration file. By default the **get-community-name** is set to **public** in the file. For details on this configuration file see the *snmpd.conf*(4) manual page.

By default, the agent does not allow managers to alter **MIB** values (it returns errors for **SNMP** SetRequests). To configure the agent to respond to **SNMP** SetRequests (AND GetRequests), add a **set-community-name** to the file **/etc/SnmpAgent.d/snmpd.conf**.

SNMPv2c

Simple Network Management Protocol Community based Version 2 (**SNMPv2**) is supported in this version of the **SNMP** Agent.

Traps

The agent also sends information to a manager without an explicit request from the manager. Such an operation is called a **trap**. By default, **SNMP** traps are not sent to any destination. To configure the agent to send traps to one or more specific destinations, add the trap destinations to **/etc/SnmpAgent.d/snmpd.conf**.

Then Master Agent (**snmpdm**) and the MIB-2 subAgent (**mib2agt**) collaborate to send the following **SNMP** traps:

coldStart Sends a coldStart trap when the SNMP Agent is invoked.

linkDown Sends a linkDown trap when an interface goes down.

linkUp Sends a linkUp trap when an interface comes up.

authenticationFailure
Sends an authenticationFailure trap when an **SNMP** request is sent to the agent with a community name that does not match any community names the agent is configured to work with.

SNMP Agent Logging

The SNMP Agent provides the capability to log various types of errors and events. There are three types of logging: Errors, Warnings and Traces.

Log Masks

Log masks enable the user to specify the particular classes of messages that should be logged to **/var/adm/snmpd.log** or *logfile*. There are three different ways in which you can specify the logmask that you want. They are: (1) decimal number, (2) hex number, or (3) text string. The three may not be used in combination.

To select multiple output types do the following. For decimal or hex format simply add the individual log-mask values together and enter that number. When entering strings, place multiple strings on the same line, space separated, without quotes.

FUNCTION	LOG MASK VALUES Decimal	Hex	String
Log factory trace messages	8388608	0x00800000	FACTORY_TRACE
Log factory warning messages	268435456	0x10000000	FACTORY_WARN
Log factory error messages	536870912	0x20000000	FACTORY_ERROR

For example, to turn on error log messages:

```
decimal format: snmpdm -m 536870912
hex     format: snmpdm -m 0x20000000
string  format: snmpdm -m FACTORY_ERROR
```

Using **-m** or **-mask** logmask command line options might cause the master agent to run in the foreground and the agent would not daemonize. This could potentially cause system hang during boot times if any of these options were added to the start up scripts, since the boot up environment might wait indefinitely for the agent to daemonize. So it is adviced not to add these command line options to the start up scripts but use these options only during an agent debug session.

Supported MIB Objects

The Management Information Base (**MIB**) is a conceptual database of values **MIB** (**objects**) on the agent. The Master SNMP Agent implements a small number of **MIB** objects but most **MIB** objects are implemented by subAgents that attach to the Master Agent. See **/opt/OV/snmp_mibs/** on systems with OpenView products installed for definitions of particular **MIB** objects.

This version of the SNMP Agent includes three subAgents, **/usr/sbin/mib2agt**, and **/usr/sbin/hp_unixagt** which implement the MIB-2 and hp-unix MIBs respectively, and the third **/usr/sbin/trapdestagt** which is used in configuring destinations for the agent's traps. The MIBs for the subagents mib2agt and hp_unixagt are described in **/opt/OV/snmp_mibs/rfc1213-MIB-II** and **/opt/OV/snmp_mibs/hp-unix** on systems with OpenView products installed.

The MIB-2 subAgent supports most of the objects in **RFC1213**. The **EGP** group is not supported. The hp-unix subAgent supports most of the objects in the hp-unix MIB.

DEPRECATED MIBS

The ieee8023Mac **MIB** group corresponding to the following OID is no longer supported:

private(4).enterprises(1).hp(11).nm(2).interface(4).ieee8023Mac(1)

This **MIB** group is replaced with the "Ether-Like" **MIB** group (RFC1398) which corresponds to OID:

mgmt(2).mib-2(1).transmission(10).dot3(7)

SNMP Agent Startup

The SNMP Agent startup mechanism is built upon the System V.4 file system paradigm. The startup script, **/etc/netmanrc**, which was used in previous releases of the SNMP Agent is no longer used.

Automatic Startup

As installed, the SNMP Master Agent and all subAgents should startup automatically each time the system re-boots or any time the system transitions from run level 1 to run level 2. When the system enters run level 2 the operating system will execute **/sbin/init.d/SnmpMaster** which will startup the Master Agent. Similarly, the operating system invoked **/sbin/init.d/SnmpMib2**, **/sbin/init.d/SnmpHpunix** and **/sbin/init.d/SnmpTrpDst** will startup the MIB2, HP Unix and Trap Dest subAgents respectively immediately after the Master Agent is started.

Prior to executing these startup scripts the system will examine all scripts in **/etc/rc.config.d** for environment variables which could potentially influence the startup of the Master Agent and each subAgent. See the particular startup script or configuration file for details on supported environment variables. The user should never modify scripts in **/sbin/init.d**. Instead the startup behavior should be controlled by adjusting values in the appropriate configuration script in **/etc/rc.config.d**.

Manual Startup

There are two ways to start the SNMP Agent manually. The first way is to execute **snmpdm** and then start each subAgent. Separate process subAgents are started by invoking the particular subAgent executables.

The second and simplest way to start the SNMP Agent manually is to execute the **snmpd** startup script which will invoke the Master Agent and all subAgents who have been installed and designed to operate in this paradigm. The **snmpd** script is layered upon the V.4 startup paradigm and so makes use of the component startup scripts in **/sbin/init.d** and configuration scripts in **/etc/rc.config.d**. When **snmpd** is invoked it starts **/usr/sbin/snmpdm**, passes all its command line arguments to it and then executes each script (S*) found in **/sbin/SnmpAgtStart.d**.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages appear. If LANG is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of LANG. If any internationalization variable contains an invalid setting, **snmpdm** behaves as if all internationalization variables are set to "C." See *environ*(5). Many SNMP Agent log messages are only available in English.

International Code Set Support

Supports single-byte character code sets.

AUTHOR

snmpd was developed by HP, Massachusetts Institute of Technology and SNMP Research.

FILES

/usr/sbin/snmpd
/usr/sbin/snmpdm
/usr/sbin/mib2agt
/usr/sbin/hp_unixagt
/usr/sbin/trapdestagt
/etc/SnmpAgent.d/snmpd.conf
/var/adm/snmpd.log
/opt/OV/snmp_mibs/
/sbin/SnmpAgtStart.d/

SEE ALSO

snmpd.conf(4).

RFC 1155, RFC 1157, RFC 1212, RFC 1213, RFC 1231, RFC 1398.

NAME

softpower - determine if softpower hardware is installed on the system

SYNOPSIS

`/sbin/softpower`

DESCRIPTION

The **`softpower`** command determines whether a software controlled power switch is installed on the system.

RETURN VALUE

`softpower` returns the following values:

`0` Softpower hardware detected on the system.
`1` Softpower hardware was not detected on the system.
`2` The command failed because it is being run on an earlier version of HP-UX that does not support the appropriate **`sysconf`** call.

AUTHOR

`softpower` was developed by HP.

SEE ALSO

sysconf(2).

NAME

spd - set physical drive parameters on an HP SCSI disk array

SYNOPSIS

spd [**-a**] [**-c**] [**-d**] [**-f**] [**-r**] [**-x**] [**-M**] *drive device_file*

DESCRIPTION

spd changes the status of a drive on an HP SCSI disk array associated with device file *device_file*.

Options

-a	Add drive. Adds a drive to the set of drives known by the controller.
-c	Clear the warning, or "failed disk" error status. Parity checking via **scn** must be performed immediately following this operation. See WARNING.
-d	Delete drive. Deletes a drive from the set of drives known by the controller.
-f	Fail drive. Marks the drive as failed, and may place the LUN(s) residing on it in a dead or degraded state, depending on RAID level, and the presence of other failed drives in the LUN.
-r	Replace drive. Marks the drive as replaced, which instructs the controller to start reconstructing the LUN(s) associated with this replaced drive.
-x	Replace and format drive. Marks a drive as replaced, and instructs the controller to physically format the drive before starting LUN reconstruction.
-M	Force the selected option. By altering the state of individual disk mechanisms contained within a disk array, the state of configured LUN devices may also be altered. For example, if the disk array has a LUN configured into a RAID 5 configuration, and one of the disk mechanisms used to create the LUN is in a FAILED state, the disk array will be operating in a DEGRADED mode. If the user selected the FAIL DRIVE option on one of the functioning disk mechanisms of the LUN, the state of the LUN would be changed from DEGRADED to DEAD. **spd** will warn the user of any significant change in LUN state resulting from their selected option. The **-M** option suppresses these warnings and performs the selected operation.
drive	Specified in the form **c***X***i***Y* , where *X* (a decimal number) represents the SCSI channel number, and *Y* (a decimal number) represents the SCSI-ID number of the disk drive.

RETURN VALUE

spd returns the following values:

0	Successful completion.
-1	Command failed (an error occurred).

DIAGNOSTICS AND ERRORS

Errors can originate from problems with:

- **spd**
- SCSI (device level) communications
- system calls

Error messages generated by spd:

usage: spd <-a | -c | -d | -f | -r | -x> <-M> <cXiY> <special>

An error in command syntax has occurred. Enter command again with all required arguments, in the order shown.

spd: device busy

To ensure that **spd** does not modify a disk array that is being used by another process, **spd** attempts to obtain exclusive access to the disk array. If the disk array is already opened by another process (for example, LVM — the Logical Volume Manager), a "**device busy**" error message is returned by the driver. To eliminate the "**device busy**" condition, determine what process has the device open. In the case of LVM, it is necessary to deactivate the volume group containing the array before configuring the array (see *vgchange*(1M)).

spd: Arg out of range

One of the arguments has exceeded its maximum or minimum size, or is incorrect in form. Check the

S

size and form of each argument.

spd: LUN does not exist
The addressed LUN is not configured, and thus is not known to the array controller.

spd: LUN # too big
The LUN number, which is derived from the device file name, is out of range.

spd: Not a raw file
Utilities must be able to open the device file for raw access.

spd: Not an HP SCSI disk array
The device being addressed is not an HP SCSI disk array.

SCSI (device level) communication errors:
Sense data associated with the failed operation is printed.

spd: Transfer length error
The amount of data actually sent to or received from the device was not the expected amount.

Error messages generated by system calls:
spd uses the following system calls:

malloc(), **free()**, **stat()**, **open()**, **close()**, **read()**, **write()**, and **ioctl()**.

Documentation for these HP-UX system calls contains information about the specific error conditions associated with each call. **spd** does not alter the value of **errno**. The interpretation of **errno** for printing purposes is performed by the system utility **strerror()**.

EXAMPLES

The following command clears the FAILED state of the drive at ID 3 on channel 2 on the HP SCSI disk array **/dev/rdsk/c2t6d0** on a series 700. The **-M** option must be selected to suppress warning messages. The **scn** operation must be performed immediately to ensure accurate data parity information.

```
spd -c -M c2i3 /dev/rdsk/c2t6d0
scn /dev/rdsk/c2t6d0
```

To add the drive at ID 3 on channel 2 to the set of drives the array controller knows about on the HP SCSI disk array **/dev/rdsk/c2t6d0** on a series 700:

```
spd -a c2i3 /dev/rdsk/c2t6d0
```

To delete the drive at ID 3 on channel 2 from the set of drives the array controller knows about on the HP SCSI disk array **/dev/rdsk/c2t6d0** on a series 800:

```
spd -d c2i3 /dev/rdsk/c2t6d0
```

To mark the drive at ID 3 on channel 2 as failed, thus rendering any redundant RAID mode LUN (s) residing on it as dead or degraded, on the HP SCSI disk array **/dev/rdsk/c2t6d0** on a series 700:

```
spd -f -M c2i3 /dev/rdsk/c2t6d0
```

To mark the drive at ID 3 on channel 2 as replaced, thus initiating the reconstruction of any redundant RAID mode LUN (s) residing on it, on the HP SCSI disk array **/dev/rdsk/c2t6d0** on a series 700:

```
spd -r b2a3 /dev/rdsk/c2t6d0
```

NOTE

Failing a drive on a RAID_0 LUN will leave it with an "optimal" LUN status, even though the controller will no longer access the failed drive and its data.

WARNING

Clearing a "failed" disk status might leave the array with inconsistent parity. This can lead to corrupted data if the array LUN ever operates in "degraded" state. Parity scan and repair must be performed immediately after clearing the "failed" state of a disk array.

DEPENDENCIES

The HP C2425 and HP C2427 disk arrays are only supported on Series 700 systems running HP-UX version 9.0X.

The HP C2430 disk array is supported on Series 700 and 800 systems running HP-UX versions 9.0X and 10.0X.

AUTHOR

spd was developed by HP.

NAME

spray - spray packets

SYNOPSIS

/usr/sbin/spray *host* [**-c** *count*] [**-l** *length*]

DESCRIPTION

spray sends a one-way stream of packets to *host* using RPC, then reports how many were received by *host* and what the transfer rate was. The host name can be either a name or an internet address.

Options

spray recognizes the following options and command-line arguments:

-c *count* Specifies how many packets to send. The default value of *count* is the number of packets required to make the total stream size 100 000 bytes.

-l *length* The number of bytes in the Ethernet packet holding the RPC call message. Since the data is encoded using XDR, and XDR only deals with 32-bit quantities, not all values of *length* are possible. The *spray* command rounds up to the nearest possible value. When *length* is greater than the size of an Ethernet packet, the system breaks the datagram into multiple Ethernet packets. The default value of *length* is 86 bytes (the size of the RPC and UDP headers).

AUTHOR

spray was developed by Sun Microsystems, Inc.

SEE ALSO

ping(1M), sprayd(1M).

NAME

sprayd - spray server

SYNOPSIS

/usr/lib/netsvc/spray/rpc.sprayd [**-l** *log_file*] [**-e**|**-n**]

DESCRIPTION

sprayd is an RPC server that records the packets sent by **spray** from another system (see *spray*(1M)).

inetd invokes **sprayd** through **/etc/inetd.conf** (see *inetd*(1M)).

Options

sprayd recognizes the following options and command-line arguments:

-l *log_file* Log any errors to the named log file, *log_file*. Errors are not logged if the **-l** option is not specified.

Information logged to the file includes date and time of the error, host name, process id and name of the function generating the error, and the error message. Note that different services can share a single log file since enough information is included to uniquely identify each error.

-e Exit after serving each RPC request. Using the **-e** option, the **inetd** security file **/var/adm/inetd.sec** can control access to RPC services.

-n Exit only if

- **portmap** dies (see *portmap*(1M)),
- Another **rpc.sprayd** registers with **portmap**, or
- **rpc.sprayd** becomes unregistered with *portmap*.

The **-n** option is more efficient because a new process is not launched for each RPC request. **-n** is the default.

AUTHOR

sprayd was developed by Sun Microsystems, Inc.

SEE ALSO

inetd(1M), spray(1M), portmap(1M), inetd.conf(4), inetd.sec(4), services(4).

NAME

sss - set spindle sync state of drives in an HP SCSI disk array

SYNOPSIS

sss -d [*drive_list*] *device_file*

sss -on [**-s**] [*drive_list*] *device_file*

sss -off [*drive_list*] *device_file*

DESCRIPTION

sss displays or changes the spindle synchronization state of the disk drives in the HP SCSI disk array associated with device file *device_file*. Though *device_file* is the name of a device file corresponding to a LUN, **sss** operates (by default) on all disk drives physically connected to the array controller, without regard to the drives' LUN ownership. Even if multiple LUNs (or sub-LUNs) are present, **sss** should be directed to only one of them (that is, specify the name of the device file for only one of the LUNs in the **sss** command line). To affect a subset of the physical drives in the array, specify which drives to affect in *drive_list*.

Options

-d Display only. Displays the current spindle synchronization status. This has two components: the drive's master or slave status and its state of spindle synchronization (on or off).

-on Sync on. Enables spindle synchronization; one drive is designated master and the rest are designated slaves, unless the **-s** "slave only" tag is present, in which case all designated drives will be slaves. If only one drive is designated, it will be a master.

-off Sync off. Disables spindle synchronization.

-s Slave only. Only used with the **-on** option. Make all designated drives slaves. This is useful when replacing a drive in a set of drives which already have spindle synchronization enabled. If you have replaced the master drive, use the **-on** option without **-s**, and specify the new drive only.

drive_list A list of drives used to specify which drives in the array will be affected by the synchronization operation. *drive_list* is in the form **c***X***i***Y* , where *X* (a decimal number) represents the SCSI channel number, and *Y* (a decimal number) represents the SCSI-ID number of the desired drive. Drives names in *drive_list* are separated by commas. If no *drive_list* is present, **sss** defaults to all physical drives attached to the array controller, regardless of which LUNs they belong to.

RETURN VALUE

sss returns the following values:

0 Successful completion.
-1 Command failed (an error occurred).

DIAGNOSTICS AND ERRORS

Errors can originate from problems with:

- **sss**
- SCSI (device level) communications
- system calls

Error messages generated by sss:

usage: sss <-d | -on [-s] | -off> [cXiY,...] <special>

sss encountered an error in command syntax. Enter the command again with the required arguments, in the order shown.

sss: Arg out of range

One of the arguments has exceeded its maximum or minimum size, or is incorrect in form. Check the size and form of each argument.

sss: device busy

To ensure that **sss** does not modify a disk array that is being used by another process, **sss** attempts to obtain exclusive access to the disk array. If the disk array is already opened by another process (for example, LVM — the Logical Volume Manager), a "**device busy**" error message is returned by the driver. To eliminate the "**device busy**" condition, determine what process has the device open. In the case of LVM, it is necessary to deactivate the volume group containing the array before configuring the spindle sync state of the drives in the array (see *vgchange*(1M)).

sss: LUN # too big
The LUN number, which is derived from the device file name, is out of range.

sss: Not a raw file
sss must be able to open the device file for raw access.

sss: Not an HP SCSI disk array
The device being addressed is not an HP SCSI disk array.

sss: Transfer length error
The amount of data actually sent to or received from the device was not the expected amount.

SCSI (device level) communication errors:
Sense data associated with the failed operation is printed.

Error messages generated by system calls:
sss uses the following system calls:

malloc(), **free()**, **stat()**, **open()**, **close()**, **read()**, **write()**, and **ioctl()**.

Documentation for these HP-UX system calls contains information about the specific error conditions associated with each call. **sss** does not alter the value of **errno**. The interpretation of **errno** for printing purposes is performed by the system utility **strerror()**.

EXAMPLES

To display the spindle synchronization status of drives on HP SCSI disk array **/dev/rdsk/c22d0s2** on a Series 800:

```
sss -d /dev/rdsk/c22d0s2
```

To enable spindle synchronization on all drives of the HP SCSI disk array **/dev/rdsk/c410d3l3s0** on a Series 700:

```
sss -on /dev/rdsk/c410d3l3s0
```

The drive on SCSI channel 3 at SCSI ID **0** of the HP SCSI disk array **/dev/rdsk/c410d3l3s0** has just been replaced. The other drives in the array are synchronized, and the replaced one was a slave. To enable spindle synchronization on the new drive on a Series 700:

```
sss -on -s c3i0 /dev/rdsk/c410d3l3s0
```

If, in the replacement scenario above, the replaced drive was the master, to enable spindle synchronization and make the new drive a master:

```
sss -on c3i0 /dev/rdsk/c410d3l3s0
```

or, alternatively, enable the whole set again:

```
sss -on /dev/rdsk/c410d3l3s0
```

DEPENDENCIES

This utility is currently supported only on HP C2425, HP C2427, and HP C2430 disk arrays.

AUTHOR

sss was developed by HP.

NAME

st - shared tape administration

SYNOPSIS

st -f *device_file* [**-r**] [**-s**]

DESCRIPTION

The **st** command provides users with a command-line interface to check the status of a shared tape device or to reclaim a shared tape device from a host system that has failed while holding a reservation on the shared tape device. The **st** command can also be used for the same purpose on shared library robotic devices. To use the **st** command you must have root user id.

Please see examples below for usage.

Options

st recognizes the following options and arguments:

-f *device_file* Specifies the tape device file or sctl pass-through device file for the shared tape/library device. This parameter is mandatory and **st** will report an error if **-f** *device_file* is omitted.

-r Allows the user to reclaim a shared tape device or shared library robotic device in the case where a host failed while holding a reservation on the shared device. This option causes a bus device reset to be issued to the device specified by the **-f** option.

-s Prints out the current status of the shared tape/library device specified by the **-f** option.

RETURN VALUE

st returns 0 upon successful completion and 1 otherwise.

EXAMPLES

```
st -f /dev/scsi/st_device -s
```

The following shows three examples of output from the above command.

```
Device is reserved by another host
```

The above output indicates that the shared device is reserved by another host and is therefore unavailable at this time.

```
Device not ready
```

The above output indicates that the shared device is not ready for use at this time.

```
Device is OK and available
```

The above output indicates that the shared device is ready for use at this time.

To reclaim a shared tape/library device from a failed host, the following command can be used:

```
st -f /dev/scsi/st_device -r
```

WARNINGS

The **-r** option must be used with care. When reclaiming devices, it must be ensured that the host from which the device is being reclaimed has in fact failed, as data may be lost as the result of reclaiming a device that is currently in use by another host.

DEPENDENCIES

The sctl SCSI pass-through driver must be configured in the kernel before this command can be used to administer the shared tape device. See *scsi_ctl*(7).

AUTHOR

st was developed by Hewlett-Packard.

SEE ALSO

mt(1), scsi(7), scsi_ctl(7).

NAME

statd - network status monitor

SYNOPSIS

/usr/sbin/rpc.statd [**-l** *log_file*]

DESCRIPTION

statd is an RPC server. It interacts with **lockd** to provide crash and recovery functions for the locking services on NFS (see *lockd*(1M)).

Options

statd recognizes the following options and command-line arguments:

-l *log_file* Log any errors to the named log file, *log_file*. Errors are not logged if the **-l** option is not specified.

Information logged to the file includes date and time of the error, host name, process id and name of the function generating the error, and the error message.

FILES

/var/statmon/sm/*
/var/statmon/sm.bak/*
/var/statmon/state

WARNINGS

Changes in status of a site are detected only upon startup of a new status monitor and lock daemon.

AUTHOR

statd was developed by Sun Microsystems, Inc.

SEE ALSO

fcntl(2), lockf(2), signal(2), lockd(1M), sm(4).

NAME

strace - write STREAMS event trace messages to standard output

SYNOPSIS

strace [*mod sub pri* **] ...**

DESCRIPTION

strace gets STREAMS event trace messages from STREAMS drivers and modules via the STREAMS log driver (**strlog(7)**), and writes these messages to standard output. By default, **strace** without arguments writes all STREAMS trace messages from all drivers and modules. **strace** with command-line arguments limits the trace messages received.

The arguments, which must be specified in groups of three, are:

mod	Specifies the STREAMS module identification number from the streamtab entry.
sub	Specifies a subidentification number (often corresponding to a minor device).
pri	Specifies a tracing priority level. **strace** gets messages of a level equal to or less than the value specified by *pri*. Only positive integer values are allowed.

The value **all** can be used for any argument in the **strace** command line to indicate that there are no restrictions for that argument.

Multiple sets of the three arguments can be specified to obtain the messages from more than one driver or module.

Only one **strace** process can open the STREAMS log driver at a time.

When **strace** is invoked, the log driver compares the sets of command line arguments with actual trace messages, returning only messages that satisfy the specified criteria.

STREAMS event trace messages have the following format:

seq time tick pri ind mod sub text

Components are interpreted as follows:

seq	Trace event sequence number.
time	Time the message was sent expressed in *hh:mm:ss*.
tick	Time the message was sent expressed in machine ticks since the last boot.
pri	Tracing priority level as defined by the STREAMS driver or module that originates the messages.
ind	Can be any combination of the following three message indicators:

E	The message has also been saved in the error log.
F	The message signaled a fatal error.
N	The message has also been mailed to the system administrator.

mod	Module identification number of the trace message source.
sub	Subidentification number of the trace message source.
text	Trace message text.

strace runs until terminated by the user.

EXAMPLES

Display all trace messages received from the driver or module identified by *mod* **28**:

```
strace 28 all all
```

Display trace messages of any tracing priority level from the driver or module identified by *mod* **28** and its minor devices identified by the *sub* **2**, **3**, or **4**:

```
strace  28 2 all  28 3 all  28 4 all
```

Display the trace messages from the same driver or module and *subs* but limit the priority levels to 0 for *subs* 2 and 3; 1 for *sub* 4, driver or module **28**:

S

```
        strace  28 2 0  28 3 0  28 4 1
```

WARNINGS

Running **strace** with several sets of arguments can impair STREAMS performance, particularly for those modules and drivers that are sending the messages.

Also be aware that **strace** may not be able to handle a large number of messages. If drivers and modules return messages to **strace** too quickly, some may be lost.

FILES

/usr/lib/nls/msg/C/strace.cat	NLS catalog for **strace**.

SEE ALSO

strclean(1M), strerr(1M), strlog(7).

NAME

strchg, strconf - change or query stream configuration

SYNOPSIS

strchg -h *module1*[**,** *module2*]...

strchg -p [**-a**|**-u** *module*]

strchg -f *file*

strconf

strconf -t

strconf -m *module*

DESCRIPTION

The **strchg** and **strconf** commands are used to change or query the configuration of the stream associated with the user's standard input. The **strchg** command pushes modules on and/or pops modules off the stream. The **strconf** command queries the configuration of the stream. Only the superuser or owner of a STREAMS device may alter the configuration of that stream.

strchg Options

The **strchg** command uses the following options:

-h *module1*[,*module2*] ...
: **strchg** pushes modules onto a stream. The modules are pushable STREAMS modules as defined by *module1*, *module2*, and so on. The modules are pushed in order. That is, *module1* is pushed first, *module2* is pushed second, etc.

-p
: With the **-p** option alone, **strchg** pops the topmost module from the stream.

-a
: With the **-p** and **-a** options, all the modules above the topmost driver are popped.

-u *module*
: With the **-p** and **-u** *module* options, all modules above but not including module are popped off the stream.

The **-a** and **-u** options are mutually exclusive.

-f *file*
: The user can specify a *file* that contains a list of modules representing the desired configuration of the stream. Each module name must appear on a separate line where the first name represents the topmost module and the last name represents the module that should be closest to the driver. The **strchg** command will determine the current configuration of the stream and pop and push the necessary modules in order to end up with the desired configuration.

The **-h**, **-f**, and **-p** options are mutually exclusive.

strconf Options

Invoked without any arguments, **strconf** prints a list of all the modules in the stream as well as the topmost driver. The list is printed in one name per line where the first name printed is the topmost module on the stream (if one exists) and the last item printed is the name of the driver.

The **strconf** command uses the following options:

-t
: Only the topmost module (if one exists) is printed.

-m *module*
: **strconf** checks if the named *module* is present on the stream. If so, **strconf** prints the message, **yes**, and returns zero. If not, **strconf** prints the message, **no**, and returns a non-zero value.

The **-t** and **-m** options are mutually exclusive.

Notes

If the user is neither the owner of the stream nor the superuser, the **strchg** command will fail. If the user does not have read permissions on the stream and is not the superuser, the **strconf** command will fail.

If modules are pushed in the wrong order, one could end up with a stream that does not function as expected. For ttys, if the line discipline module is not pushed in the correct place, one could have a terminal that does not respond to any commands.

DIAGNOSTICS

strchg returns zero on success. It prints an error message and returns non-zero status for various error conditions, including usage error, bad module name, too many modules to push, failure of an **ioctl** on the stream, or failure to open file from the **-f** option.

strconf returns zero on success (for the **-m** or **-t** option, "success" means the named or topmost module is present). It returns a non-zero status if invoked with the **-m** or **-t** option and the module is not present. It prints an error message and returns non-zero status for various error conditions, including usage error or failure of an **ioctl** on the stream.

EXAMPLES

The following command pushes the module **ldterm** on the stream associated with the user's standard input:

```
strchg -h ldterm
```

The following command pops the topmost module from the stream associated with **/dev/term/24**. The user must be the owner of this device or be superuser.

```
strchg -p < /dev/term/24
```

If the file, **fileconf**, contains the following:

```
compat
ldterm
ptem
```

then the command

```
strchg -f fileconf
```

will configure the user's standard input stream so that the module **ptem** is pushed over the driver, followed by **ldterm** and **compat** closest to the stream head.

The **strconf** command with no arguments lists the modules and topmost driver on the stream. For a stream that only has the module **ldterm** pushed above the ports driver, it would produce the following output:

```
ldterm
ports
```

The following command asks if **ldterm** is on the stream:

```
strconf -m ldterm
```

and produces the following output while returning an exit status of 0:

```
yes
```

FILES

/usr/lib/nls/msg/C/strchg.cat	NLS catalogs
/usr/lib/nls/msg/C/strconf.cat	NLS catalogs

SEE ALSO

streamio(7).

NAME

strclean - remove outdated STREAMS error log files

SYNOPSIS

strclean [**-d** *logdir*] [**-a** *age*]

DESCRIPTION

strclean cleans the STREAMS error logger directory of log files (**error.***mm-dd*) that contain error messages sent by the STREAMS log driver, *strlog*(7). If the **-d** option is not used to specify another directory, **strclean** removes error log files in the **/var/adm/streams** directory. If the **-a** option is not used to specify another age, **strclean** removes error log files that have not been modified in three days.

Options

strclean recognizes the following options and command-line arguments:

-d *logdir*	Specifies a directory for the location of the STREAMS error log files to be removed if this is not the default directory **/var/adm/streams**.
-a *age*	Specifies a maximum age in days for the STREAMS error log files if this not the default age of 3. The value of *age* must be an integer greater than or less than 3.

EXAMPLES

Remove day-old error log files from a directory called **/tmp/streams**:

```
strclean -d /tmp/streams -a 1
```

FILES

/var/adm/streams/error.mm-dd	One or more error log file or files on which **strclean** operates. The *mm-dd* in the filename indicates the month and day of the messages contained in the file.
/usr/lib/nls/msg/C/strclean.cat	NLS catalog for **strclean**.

SEE ALSO

strerr(1M), strlog(7).

NAME

strdb - STREAMS debugging tool

SYNOPSIS

strdb [*system*]

DESCRIPTION

strdb symbolically displays the contents of various STREAMS data structures. The argument *system* allows substitutes for the default **/stand/vmunix**. **strdb** can only handle a 32-bit kernel actually running on a system. For crash dumps and for 64-bit kernels, use the **q4** debugger.

strdb runs in two modes, STREAMS subsystem and primary. STREAMS subsystem commands report the status of open streams. Primary commands display STREAMS data structures.

In a typical **strdb** session, you will do the following:

- Run **strdb**. When **strdb** starts up, you are in primary mode.
- Execute the **:S** command to enter STREAMS subsystem mode.
- Enter STREAMS subsystem commands such as **s**, **d**, and **la** to find the open stream you want to examine.
- Enter the **qh** command to select a stream and display the stream head read queue. This command returns you to primary mode.
- Enter primary mode navigation keys to display fields in the stream head read queue, and traverse the rest of the stream's queues.

The following commands are available in primary mode.

:?	Display a help menu for primary mode.
^D	Exit from **strdb**.
:q	Exit from **strdb**.
^K	If logging is enable, dump current screen to log file
^L	Refresh the screen.
:u	Disable data structure stacking. By default data structure stacking is turned on. When stacking is on, **strdb** pushes each structure it displays onto a stack so that it can be reviewed later. See **^P** and other stack commands described below.
:s	Re-enable data structure stacking. By default data structure stacking is turned on. The **:u** command turns it off. When stacking is on, **strdb** pushes each structure it displays onto a stack so that it can be reviewed later. See **^P** and other stack commands described below.
:l *name* **o**|**c**	If the **o** option is specified, open a log file, *name*, and start logging. Alternatively if the **c** option is specified, close a log file, *name*, and stop logging.
:S	Enter STREAMS subsystem mode.
navigation key	Display the field specified by *navigation key* in the currently displayed data structure. **strdb** provides different navigation keys for each STREAMS data structure. Each key indicates a particular field in the data structure to display. The navigation keys for STREAMS data structures are described after the STREAMS subsystem commands below.
?	Display a help menu for the displayed data structure's navigation keys.
^R	Update the displayed data structure with new values from **/dev/kmem** on a running system.
^P	Pop the displayed data structure off the data structure stack, and display the data structure now at the top of the stack.
:m	Mark the displayed data structure. Later the data structure stack can be popped back to this structure using **^U** as described below.
^U	Pop the data structure stack back to a structure marked with **:m**, and display this structure.

S

^T	Transpose the top two data structure stack entries. Unlike **^P**, this command allows the data structure on the top of the stack to be saved for later viewing.
:b *addr* [*len*]	Display *len* bytes of binary data at address *addr*. The default for *len* is 256.
:x *name addr*	Display structure *name* located at address *addr*.
:x ?	Display the structure names accepted by **:x**.

The following commands are available in STREAMS subsystem mode.

?	Display a help menu for STREAMS subsystem mode.
h	Display a help menu for STREAMS subsystem mode.
q	Exit from STREAMS subsystem mode to primary mode.
v	Print the version of STREAMS data structures displayed.
s d | **m**	If the **d** option is specified, list the STREAMS drivers included in the kernel **S800** or **dfile** file. Alternatively if the **m** option is specified, list the included modules.
la *name*	List all open streams for device *name*. The *name* is one of those shown by the **s** command.
lm *name minor*	List all modules pushed on the stream for device *name* and minor *minor*.
ll *name minor*	List all drivers linked under the multiplexor *name* with minor *minor*.
lp *name minor*	List all drivers persistently linked under the multiplexor *name* with minor *minor*.
qc *name file*	Write the **q_count** values for the driver *name* into file, *file*.
qh *name minor*	Display the streams head read queue for the STREAMS driver *name* and minor *minor*. This command returns the user to primary mode.

strdb provides different navigation keys for each STREAMS data structure it displays. Each key indicates a particular data structure field to display. The navigation keys for each STREAMS data structure are described in the following paragraphs.

The navigation keys for the STREAMS **queue** structure are:

i	Displays the **q_init** structure pointed to by the **q_qinfo** field.
m	Displays the **msgb** structure pointed to by the **q_first** field.
z	Displays the **msgb** structure pointed to by the **q_last** field.
n	Displays the **queue** structure pointed to by the **q_next** field.
l	Displays the **queue** structure pointed to by the **q_link** field.
b	Displays the **qband** structure pointed to by the **q_bandp** field.
o	Displays the **queue** structure pointed to by the **q_other** field.

The navigation keys for the STREAMS **qinit** structure are:

i	Displays the **module_info** structure pointed to by the **qi_minfo** field.
s	Displays the **module_stat** field pointed to by the **qi_mstat** field.

The navigation keys for the STREAMS **msgb** structure are:

n	Displays the **msgb** structure pointed to by the **b_next** field.
p	Displays the **msgb** structure pointed to by the **b_prev** field.
m	Displays the data pointed to by the **b_rptr** field.
c	Displays the **msgb** structure pointed to by the **b_cont** field.
d	Displays the **datab** structure pointed to by the **b_datap** field.

The navigation keys for the STREAMS **datab** structure are:

d	Displays the **a__datab** structure pointed to by the **db_f** field.

The navigation keys for the STREAMS **qband** structure are:

n	Displays the **qband** structure pointed to by the **qb_next** field.
f	Displays the **msgb** structure pointed to by the **qb_first** field.
l	Displays the **msgb** structure pointed to by the **qb_last** field.

AUTHOR

strdb was developed by HP.

SEE ALSO

STREAMS/UX for HP9000 Reference Manual.

NAME

strerr - receive error messages from the STREAMS log driver

SYNOPSIS

strerr [**-a** *sys_admin_mail_name*] [**-d** *logdir*]

DESCRIPTION

The **strerr** daemon receives error messages from the STREAMS log driver (*strlog*(7)) for addition to the STREAMS error log files (**error.***mm-dd*) in the STREAMS error logger directory (**/var/adm/streams** by default). When first called, **strerr** creates the log file **error.***mm-dd*. This is a daily log file, where *mm* indicates the month and *dd* indicates the day of the logged messages. **strerr** then appends error messages to the log file as they are received from the STREAMS log driver.

STREAMS error log messages have the following format:

seq time tick pri ind mod sub text

Components are interpreted as follows:

seq	Error event sequence number.
time	Time the message was sent expressed in *hh:mm:ss*.
tick	Time the message was sent expressed in machine ticks since the last boot.
pri	Error priority level as defined by the STREAMS driver or module that originates the messages.
ind	Can be any combination of the following three message indicators:

T	The message has also been saved in the trace log.
F	The message signaled a fatal error.
N	The message has also been mailed to the system administrator.

mod	Module identification number of the error message source.
sub	Subidentification number of the error message source.
text	Error message text.

strerr runs continuously until terminated by the user.

Options

strerr recognizes the following options and command-line arguments:

-a *sys_admin_mail_name*	Specify the user's mail name for sending mail messages. Mail is sent to the system administrator by default.
-d *logdir*	Specify the directory to contain the error log file. Default is **/var/adm/streams**.

S

WARNINGS

Only one **strerr** process can open the STREAMS log driver at a time. This restriction is intended to maximize performance.

The STREAMS error logging mechanism works best when it is not overused. **strerr** can degrade STREAMS performance by affecting the response, throughput, and other behaviors of the drivers and modules that invoke it. **strerr** also fails to capture messages if drivers and modules generate messages at a higher rate than its optimum read rate. If there are missing sequence numbers among the messages in a log file, messages have been lost.

FILES

/usr/lib/nls/msg/C/strerr.cat	NLS catalog for **strerr**.
/var/adm/streams/error.mm-dd	error log file or files on which **strerr** operates

SEE ALSO

strace(1M), strlog(7).

NAME

strvf - STREAMS verification tool

SYNOPSIS

strvf [**-v**]

DESCRIPTION

strvf executes a series of subcommands that verify whether or not STREAMS is currently installed and configured on your system. All output is sent to **stdout**. Verbose output is always sent to the logfile **/var/adm/streams/strvf.log**.

These subcommands make sure that the STREAMS kernel daemons are running and that **open()**, **putmsg()**, **getmsg()**, **ioctl()**, and **close()** can be performed on **/dev/echo**.

Options

-v	Specifies verbose output to be displayed

EXAMPLES

strvf	Verify STREAMS is working. Brief summary of status is displayed on screen. Verbose description of each subcommand and its status is copied to the logfile.
strvf -v	Verify STREAMS is working. Verbose description of each subcommand and its status is displayed on the screen and copied to the logfile. This option is useful in troubleshooting **strvf** failures.

FILES

/var/adm/streams/strvf.log	Logfile containing a verbose description and status of all subcommands.
/dev/echo	Loopback STREAMS driver used by **strvf**.

SEE ALSO

open(2), close(2), getmsg(2), putmsg(2), streamio(7).

NAME

swacl - view or modify the Access Control Lists (ACLs) which protect software products

SYNOPSIS

swacl -l *level* [**-D** *acl_entry* | **-F** *acl_file* | **-M** *acl_entry*] [**-f** *software_file*] [**-t** *target_file*] [**-x** *option=value*] [**-X** *option_file*] [*software_selections*] [**@** *target_selections*]

Remarks

- This command supports operations on remote systems. See the **Remote Operation** section below for details.
- Type **man 5 sd** to display *sd*(5) for an overview of all SD commands.

DESCRIPTION

The **swacl** command displays or modifies the Access Control Lists (ACLs) which:

- Protect the specified *target_selections* (hosts, software depots or root filesystems).
- Protect the specified *software_selections* on each of the specified *target_selections* (software depots only).

All root filesystems, software depots, and products in software depots are protected by ACLs. The SD commands permit or prevent specific operations based on whether the ACLs on these objects permit the operation. The **swacl** command is used to view, edit, and manage these ACLs. The ACL must exist and the user must have the appropriate permission (granted by the ACL itself) in order to modify it.

ACLs offer a greater degree of selectivity than standard file permissions. ACLs allow an object's owner (i.e. the user who created the object) or the local superuser to define specific read, write, or modify permissions to a specific list of users, groups, or combinations thereof.

Some operations allowed by ACLs are run as local superuser. Because files are loaded and scripts are run as superuser, granting a user write permission on a root filesystem or insert permission on a host effectively gives that user superuser privileges.

Protected Objects

The following objects are protected by ACLs:

- Each host system on which software is being managed by SD,
- Each root filesystem on a host (including alternate roots),
- Each software depot on a host,
- Each software product contained within a depot.

Remote Operation

You can enable SD to manage software on remote systems. To let the root user from a central SD *controller* (also called the *central management server* or *manager node*) perform operations on a remote *target* (also called the *host* or *agent*):

1)

Install a special HP ServiceControl Manager fileset on the remote systems. This enables remote operations by automatically setting up the root, host, and template Access Control Lists (ACLs) on the remote machines and permitting root access from the controller system. To install the fileset, run the following command on each remote system:

swinstall -s *controller* **:/var/opt/mx/depot11 AgentConfig.SD-CONFIG**

NOTES:

- *controller* is the name of the central management server.
- If the target is running HP-UX 10.20, use the same command but substitute **depot10** for **depot11**.
- Targets previously set up by SD/OV to be managed by this controller do not need this step.
- SD does not require any other ServiceControl Manager filesets.

2)

(Optional) **swinstall**, **swcopy**, and **swremove** have enhanced GUI interfaces for remote operations. Enable the enhanced GUIs by creating the **.sdkey** file on the controller. Use this command:

touch /var/adm/sw/.sdkey

NOTES:

- This step is not required when you use SD from within the HP ServiceControl Manager.
- See *sd*(5), *swinstall*(1M), *swcopy*(1M), *swjob*(1M), *swlist*(1M) or *swremove*(1M) for more information on interactive operations.

NOTE: You can also set up remote access by using **swacl** directly on the remote machines to grant root or non-root access to users from the controller system.

Options

If the **-D**, **-F**, or **-M** option is not specified, **swacl** prints the requested ACL(s) to the standard output.

The **swacl** command supports the following options:

-D *acl_entry* Deletes an existing entry from the ACL associated with the specified object(s). For this option, the permission field of the ACL entry is not required. You can specify multiple **-D** options. See the **ACL Entries** heading for more information.

-f *software_file*
Read the list of *software_selections* from *software_file* instead of (or in addition to) the command line.

-F *acl_file* Assigns the ACL contained in *acl_file* to the object. All existing entries are removed and replaced by the entries in the file. Only the ACL's entries are replaced; none of the information contained in the comment portion (lines with the prefix "#") of an ACL listing is modified with this option. The *acl_file* is usually the edited output of a **swacl** list operation.

If the replacement ACL contains no syntax errors and the user has **control** permission on the ACL (or is the local superuser), the replacement succeeds.

-l *level* Defines which level of SD ACLs to view/modify.

The supported *levels* of depot, host, root, and product objects that can be protected are:

depot View/modify the ACL protecting the software depot(s) identified by the *target_selections*.

host View/modify the ACL protecting the host system(s) identified by the *target_selections*.

root View/modify the ACL protecting the root filesystem(s) identified by the *target_selections*.

product View/modify the ACL protecting the software product identified by the *software_selection*. Applies only to products in depots, not installed products in roots.

The supported *levels* of templates are:

global_soc_template
View/modify the template ACL used to initialize the ACL(s) of future software depot(s) or root filesystem(s) added to the host(s) identified by the *target_selections*. Additionally, **swacl** can create templates that you can re-use to create new ACLs.

global_product_template
View/modify the template ACL used to initialize the **product_template** ACL(s) of future software depot(s) added to the host(s) identified by the *target_selections*.

product_template
View/modify the template ACL used to initialize the ACL(s) of future product(s) added to the software depot(s) identified by the *target_selections*.

-M *acl_entry* Adds a new ACL entry or changes the permissions of an existing entry. You can specify multiple **-M** options. See the **ACL Entries** heading for more information.

-t *target_file* Read the list of *target_selections* from *file* instead of (or in addition to) the command line.

-x *option=value*
Set the session *option* to *value* and override the default value (or a value in an alternate *option_file* specified with the **-X** option). You can specify multiple **-x** options.

-X *option_file* Read the session options and behaviors from *option_file*.

You can specify only one of the **-D**, **-F**, or **-M** options at each invocation of **swacl**.

Operands

Most SD commands support two types of operands: *software selections* followed by *target selections*. These operands are separated by the "@" (at) character. This syntax implies that the command operates on "software selections at targets".

Software Selections

The **swacl** command supports the following syntax for each *software_selection*:

product[**,***version*]

- The **=** (equals) relational operator lets you specify selections with the following shell wildcard and pattern-matching notations:

  [ ], *****, **?**

- The **\*** software specification selects all products in the depot when used with **-l** *product*.

The *version* component usually has the following form:

[**,r** *<op> revision*][**,a** *<op> arch*][**,v** *<op> vendor*]
[**,c** *<op> category*]

- The *<op>* (relational operator) component can take the form:

  =, **==**, **>=**, **<=**, **<**, **>**, or **!=**

  which performs individual comparisons on dot-separated fields.

  For example, **r>=B.10.00** chooses all revisions greater than or equal to **B.10.00**. The system compares each dot-separated field to find matches. Shell patterns are not allowed with these operators.

- The **=** (equals) relational operator lets you specify selections with the shell wildcard and pattern-matching notations:

  [ ], *****, **?**, **!**

  For example, the expression **r=1[01].*** returns any revision in version 10 or version 11.

- All version components are repeatable within a single specification (e.g. **r>=A.12**, **r<A.20**). If multiple components are used, the selection must match all components.
- Fully qualified software specs include the **r=**, **a=**, and **v=** version components even if they contain empty strings.
- No space or tab characters are allowed in a software selection.
- The software *instance_id* can take the place of the version component. It has the form:

  [*instance_id*]

  within the context of an exported catalog, where *instance_id* is an integer that distinguishes versions of products and bundles with the same tag.

Target Selections

The SD commands support this syntax for each *target_selection*.

[*host*][**:**][**/***directory*]

The **:** (colon) is required if both a host and directory are specified.

EXTERNAL INFLUENCES

Default Options

In addition to the standard options, you can change SD behaviors and policy options by editing the default values found in:

/var/adm/sw/defaults	the system-wide default values,
$HOME/.swdefaults	the user-specific default values.

You must use the following syntax to specify values in the defaults file:

[*command_name*.]*option*=*value*

The optional *command_name* prefix denotes one of the SD commands. Using the prefix limits the change in the default value to that command. If you leave the prefix off, the change applies to all commands.

You can also override default values from the command line with the **-x** or **-X** options:

command **-x** *option*=*value*

command **-X** *option_file*

The following section lists all of the keywords supported by the **swacl** command. If a default value exists, it is listed after the "=".

admin_directory=/var/adm/sw (for normal mode)
admin_directory=/var/home/LOGNAME/sw (for nonprivileged mode)

The location for SD logfiles and the default parent directory for the installed software catalog. The default value is **/var/adm/sw** for normal SD operations. When SD operates in nonprivileged mode (that is, when the **run_as_superuser** default option is set to **true**):

- The default value is forced to **/var/home/LOGNAME/sw**
- The path element **LOGNAME** is replaced with the name of the invoking user, which SD reads from the system password file.
- If you set the value of this option to **HOME/***path*, SD replaces **HOME** with the invoking user's home directory (from the system password file) and resolves *path* relative to that directory. For example, **HOME/my_admin** resolves to the **my_admin** directory in your home directory.
- If you set the value of the **installed_software_catalog** default option to a relative path, that path is resolved relative to the value of this option.

SD's nonprivileged mode is intended only for managing applications that are specially designed and packaged. You cannot use this mode to manage the HP-UX operating system or patches to it. For a full explanation of nonprivileged SD, see the *Software Distributor Administration Guide*, available at the **http://docs.hp.com** web site.

See also the **installed_software_catalog** and **run_as_superuser** options.

distribution_target_directory=/var/spool/sw

Defines the default location of the target depot.

installed_software_catalog=products

Defines the directory path where the Installed Products Database (IPD) is stored. This information describes installed software. When set to an absolute path, this option defines the location of the IPD. When this option contains a relative path, the SD controller appends the value to the value specified by the **admin_directory** option to determine the path to the IPD. For alternate roots, this path is resolved relative to the location of the alternate root. This option does not affect where software is installed, only the IPD location.

This option permits the simultaneous installation and removal of multiple software applications by multiple users or multiple processes, with each application or group of applications using a different IPD.

Caution: use a specific installed_software_catalog to manage a specific application. SD does not support multiple descriptions of the same application in multiple IPDs.

See also the **admin_directory** and **run_as_superuser** options, which control SD's nonprivileged mode. (This mode is intended only for managing applications that are specially designed and packaged. You cannot use this mode to manage the HP-UX operating system or patches to it. For a full explanation of nonprivileged SD, see the *Software Distributor Administration Guide*, available at the **`http://docs.hp.com`** web site.)

`level=` Defines the level of SD ACLS to view/modify. The supported levels are: **`host`**, **`depot`**, **`root`**, **`product`**, **`product_template`**, **`global_soc_template`**, or **`global_product_template`**.

See the discussion of the **`-l`** option above for more information.

`rpc_binding_info=ncacn_ip_tcp:[2121] ncadg_ip_udp:[2121]`

Defines the protocol sequence(s) and endpoint(s) on which the daemon listens and which the other commands use to contact the daemon. If the connection fails for one protocol sequence, the next is attempted. SD supports both the tcp (ncacn_ip_tcp:[2121]) and udp (ncadg_ip_udp:[2121]) protocol sequence on most platforms.

`rpc_timeout=5`

Relative length of the communications timeout. This is a value in the range from 0 to 9 and is interpreted by the DCE RPC. Higher values mean longer times; you may need a higher value for a slow or busy network. Lower values will give faster recognition on attempts to contact hosts that are not up, or are not running **`swagentd`**. Each value is approximately twice as long as the preceding value. A value of 5 is about 30 seconds for the **`ncadg_ip_udp`** protocol sequence. This option may not have any noticeable impact when using the **`ncacn_ip_tcp`** protocol sequence.

`run_as_superuser=true`

This option controls SD's nonprivileged mode. This option is ignored (treated as true) when the invoking user is super-user.

When set to the default value of true, SD operations are performed normally, with permissions for operations either granted to a local super-user or set by SD ACLs. (See *swacl*(1M) for details on ACLs.)

When set to false and the invoking user is local and is *not* super-user, nonprivileged mode is invoked:

- Permissions for operations are based on the user's file system permissions.
- SD ACLs are ignored.
- Files created by SD have the uid and gid of the invoking user, and the mode of created files is set according to the invoking user's umask.

SD's nonprivileged mode is intended only for managing applications that are specially designed and packaged. You cannot use this mode to manage the HP-UX operating system or patches to it. For a full explanation of nonprivileged SD, see the *Software Distributor Administration Guide*, available at the **`http://docs.hp.com`** web site.

See also the **`admin_directory`** and **`installed_software_catalog`** options.

`select_local=true`

If no *target_selections* are specified, select the default **`target_directory`** of the local host as the *target_selection* for the command.

`software=`

Defines the default *software_selections*. There is no supplied default. If there is more than one software selection, they must be separated by spaces.

`targets=`

Defines the default *target_selections*. There is no supplied default (see **`select_local`** above). If there is more than one target selection, they must be separated by spaces.

`verbose=1`

Controls the verbosity of the output (stdout). A value of

`0` disables output to stdout. (Error and warning messages are always written to stderr).

`1` enables verbose messaging to stdout.

Environment Variables

SD programs are affected by external environment variables, set environment variables for use by the control scripts, and use other environment variables that affect command behavior.

The external environment variable that affects the **swacl** command is:

LANG Determines the language in which messages are displayed. If LANG is not specified or is set to the empty string, a default value of **C** is used. See the *lang*(5) man page by typing **man 5 lang** for more information.

Note: The language in which the SD agent and daemon log messages are displayed is set by the system configuration variable script, **/etc/rc.config.d/LANG**. For example, **/etc/rc.config.d/LANG**, must be set to **LANG=ja_JP.SJIS** or **LANG=ja_JP.eucJP** to make the agent and daemon log messages display in Japanese.

LC_ALL Determines the locale used to override any values for locale categories specified by the settings of **LANG** or any environment variables beginning with **LC_**.

LC_CTYPE
Determines the interpretation of sequences of bytes of text data as characters (e.g., single- versus multibyte characters in values for vendor-defined attributes).

LC_MESSAGES
Determines the language in which messages are written.

LC_TIME
Determines the format of dates (*create_date* and *mod_date*) when displayed by **swlist**. Used by all utilities when displaying dates and times in **stdout**, **stderr**, and **logging**.

TZ Determines the time zone for use when displaying dates and times.

OPERATION

ACL Entries

Each entry in an ACL has the following form:

entry_type[**:** *key*]**:** *permissions*

For example: **user:steve@newdist:crwit**

An ACL can contain multiple entries. See the **Entry Types** and **Permissions** headings below for more information.

Entry Types

The following *entry_types* are supported:

any_other Permissions for all other users and hosts that do not match a more specific entry in the ACL. (Example: **any_other:-r--t**.)

group Permissions for a named group. This type of ACL entry must include a key that identifies that group. The format can be: **group:***group_name***:***permissions* or **group:***group_name*@*hostname***:***permissions*. (Example: **group:adm:crwit**.)

host Permissions for an SD agent from the specified host system. SD agents require *product level* read access via either a **host**, **other**, or **any_other** entry type in order to copy or install products from depots. This type of ACL entry must include a key containing a hostname or number (in Internet dot notation) of a system or the asterisk character (*****) to denote all systems. (Example: **host:newdist@fc.hp.com:-r--t**.)

object_owner
Permissions for the object's owner, whose identity is listed in the comment header. (Example: **object_owner:crwit**.)

object_group
Permissions for members of the object's group, whose identity is listed in the comment header. (Example: **object_group:crwit**.)

other Permissions for others who are not otherwise named by a more specific entry type. The format for **other** can be: **other:***permissions* for others on the local host (only one such entry allowed) or **other:@***hostname***:***permissions* for others at remote hosts (Only one such entry per remote host allowed). (Example:

S

other:@newdist:-r--t.)

user Permissions for a named user. This type of ACL entry must include a key that identifies that user. The format for **user** can be: **user:***user_name***:***permissions* or **user:***user_name***@***hostname***:***permissions*. (Example: **user:rml:crwit**.)

Permissions

Permissions are represented as the single character abbreviations indicated below. Some permissions either apply only to, or have different meaning for, certain types of objects, as detailed below. The following permissions may be granted:

r *ead* Grants permission to read the object. On **host**, **depot**, or **root** objects, read permission allows **swlist** operations. On products within depots, read permission allows product files to be installed or copied with **swinstall** or **swcopy**.

w *rite* Grants permission to modify the object itself.

- On a **root** object (e.g. installed root filesystem), this also grants permission to modify the products installed (contained) within it.
- On a **depot** object, it does **not** grant permission to modify the products contained within it. Write access on products is required to modify products in a depot.
- On a **host** container, write permission grants permission to unregister depots. It does **not** grant permission to modify the depots or roots contained within it.

i *nsert* On a **host** object, grants permission to create (insert) a new software depot or root filesystem object, and to register roots and depots. On a **depot** object, grants permission to create (insert) a new product object into the **depot**.

c *ontrol* Grants permission to modify the ACL using **swacl**.

t *est* Grants permission to perform access checks and to list the ACL.

a *ll* A wildcard which grants all of the above permissions. It is expanded by **swacl** to **crwit**.

List Output Format

The output of a list operation is in the following format:

```
# swacl   Object_type      Access Control List
#
# For   depot | host:[host]:[/directory]
#
# Date:  date_stamp
#
# Object Ownership:  User=   user_name
#                    Group=  group_name
#                    Realm=  host_name
#
# default_realm = host_name
entry_type:[key:]permissions
entry_type:[key:]permissions
entry_type:[key:]permissions
```

You can save this output into a file, modified it, then use it as input to a **swacl** modify operation (see the **-F** option above).

Object Ownership

An *owner* is also associated with every SD object, as defined by the user name, group and hostname. The owner is the user who created the object. When using **swacl** to view an ACL, the owner is printed as a comment in the header.

Default Realm

An ACL defines a default *realm* for an object. The realm is currently defined as the name of the host system on which the object resides. When using **swacl** to view an ACL, the default realm is printed as a comment in the header.

Keys

Expressions (patterns) are **not** permitted in keys.

A key is required for **user**, **group** and **host** entry types. A key is optional for **other** entry types, and specifies the hostname to which the entry applies. Only one **other** entry type may exist *without* a key, and this entry applies to users at the default realm (host) of the ACL.

A hostname in a key is listed in its Internet address format (dot notation) if **swacl** cannot resolve the address using the local lookup mechanism (DNS, NIS, or */etc/hosts*). A hostname within an ACL entry must be resolvable when used with the **-D** and **-M** options. Unresolvable hostname values are accepted in files provided with the **-F** option.

RETURN VALUE

The **swacl** command returns:

0 The *software_selections* and/or *target_selections* were successfully displayed or modified.
1 The display/modify operation failed on all *target_selections*.
2 The modify/modify operation failed on some *target_selections*.

DIAGNOSTICS

The **swacl** command writes to stdout, stderr, and to the daemon logfile.

Standard Output

The **swacl** command prints ACL information to stdout when the user requests an ACL listing.

Standard Error

The **swacl** command writes messages for all WARNING and ERROR conditions to stderr. A report that the *software_selections* do not exist is also given if the user has **no** access permissions to the object.

Logging

The **swacl** command does not log summary events. It logs events about each ACL which is modified to the **swagentd** logfile associated with each *target_selection*.

EXAMPLES

To list the ACLs for the **C** and **OPENVIEW** products in depot **/var/spool/swtest**:

```
swacl -l product C OPENVIEW  @ /var/spool/swtest
```

The ACL listed to the standard output is similar to this example ACL:

```
#
# swacl   Product Access Control Lists
#
# For depot: newdist:/var/spool/swtest
#
# Date:  Wed May 26 11:14:31 2000
#
#
# For product:  OPENVIEW,r=3.2
#
#
# Object Ownership:  User=  robason
#                    Group= swadm
#                    Realm= newdist.fc.hp.com
#
# default_realm=newdist.fc.hp.com
object_owner:crwit
group:swadm:crwit
any_other:-r--t
#
# For product:  C,r=9.4
#
#
# Object Ownership:  User=  robason
#                    Group= swadm
```

S

```
#                       Realm= newdist.fc.hp.com
#
# default_realm=newdist.fc.hp.com
object_owner:crwit
user:rob@lehi.fc.hp.com:-r--t
user:barb:-r--t
user:ramon:-r--t
group:swadm:crwit
other:-r--t
host:lehi.fc.hp.com:-r--t
```

To list the product template ACL on host **newdist**:

```
swacl -l global_product_template  @ newdist
```

To list the host ACL on the local system:

```
swacl -l host
```

To read, edit, then replace the ACL protecting the default depot **/var/spool/sw**:

```
swacl -l depot > new_acl_file
vi new_acl_file
swacl -l depot -F new_acl_file
```

To allow user **allen** to create, register, and manage all new depots and roots on the local system:

```
swacl -l host -M user:allen:a
swacl -l global_soc_template -M user:allen:a
swacl -l global_product_template -M user:allen:a
```

To allow user **allen** to fully manage **my_depot**, which already exists:

```
swacl -l depot -M user:allen:a @ /my_depot
swacl -l product_template -M user:allen:a @ /my_depot
swacl -l product -M user:allen:a \* @ /my_depot
```

To deny general access to a depot:

```
swacl -l depot -M any_other:- @ /restricted_depot
swacl -l product -M any_other:- \* @ /restricted_depot
swacl -l product_template -M any_other:- @ /restricted_depot
```

To allow user **allen** on host **gemini** access to **restricted_depot** and all products it currently contains:

```
swacl -l depot -M host:*:rt @ /restricted_depot
swacl -l depot -M user:allen@gemini:rt @ /restricted_depot
swacl -l product -M user:allen@gemini:rt \* @ /restricted_depot
```

S

To revoke previously granted ACL permission for user **allen** on host **gemini** to access the **WDB** product in the default depot on **lehi**:

```
swacl -l product -D user:allen@gemini WDB @ lehi
```

To deny access to the default depot on the local system from host **numenal**:

```
swacl -l depot -M host:numenal:-
```

To deny access to the **OPENVIEW** product in the default depot on host **lehi** to all users who do not have an explicit ACL entry:

```
swacl -l product -M any_other:t OPENVIEW @ lehi
```

To allow user **george** on host **newdist** access to the **OPENVIEW** product in the default depot on host **lehi**, you must specify both a user ACL for **george** and a host ACL for **newdist**:

```
swacl -l product -M user:george@newdist:rt OPENVIEW @ lehi
swacl -l product -M host:newdist:rt OPENVIEW @ lehi
```

To revoke a user ACL for user **allen** on host **gemini** that allowed access to the **OPENVIEW** product in the default depot on host **lehi**:

```
swacl -l product -D user:allen@gemini OPENVIEW @ lehi
```

To revoke any previously issued access to the **OPENVIEW** product in the default depot on host **lehi** by users on host **numenal**:

```
swacl -l product -D host:numenal OPENVIEW @ lehi
```

To deny all access to the users **steve** and **george** for the depot **/var/spool/sw** at host **newdist**:

```
swacl -l depot -M user:steve:- -M user:george:- \
      @ newdist:/var/spool/sw
```

To delete entries for local user **rick** from all products in the default local depot:

```
swacl -l product -D user:rick \*
```

WARNINGS

- You can edit an ACL in such a way that it will leave a system inaccessible. Do not remove all **control** permissions on an ACL. (Note, however, that the local super-user can always edit SD ACLs, regardless of permissions.)
- ACLs can grant the equivalent of local superuser permission. SD loads and runs files and scripts as superuser. Therefore, if an SD ACL gives a user write permission on a root filesystem or insert permission on a host, that user has the equivalent of superuser privileges.
- Note that **swacl** is not a general purpose ACL editor. It works only on ACLs protecting SD objects.

FILES

$HOME/.swdefaults
Contains the user-specific default values for some or all SD options.

/usr/lib/sw/sys.defaults
Contains the master list of current SD options (with their default values).

/var/adm/sw/
The directory which contains all of the configurable (and non-configurable) data for SD. This directory is also the default location of logfiles.

/var/adm/sw/defaults
Contains the active system-wide default values for some or all SD options.

/var/adm/sw/products/
The Installed Products Database (IPD), a catalog of all products installed on a system.

/var/adm/sw/security/
The directory which contains ACLs for the system itself, template ACLS, and the secrets file used to authenticate remote requests.

/var/spool/sw/
The default location of a source and target software depot.

S

AUTHOR

swacl was developed by the Hewlett-Packard Company.

SEE ALSO

sd(5), sd(4), swpackage(4), swagentd(1M), swask(1M), swconfig(1M), swcopy(1M), swinstall(1M), swjob(1M), swlist(1M), swmodify(1M), swpackage(1M), swreg(1M), swremove(1M), swverify(1M), install-sd(1M).

Software Distributor Administration Guide, available at **http://docs.hp.com**.

SD customer web site at **http://software.hp.com/SD_AT_HP/**.

NAME

swagent, swagentd - serve local or remote SD software management tasks, daemon that invokes swagent, respectively

SYNOPSIS

swagent executed by **swagentd** only.

swagentd [**-k**] [**-n**] [**-r**] [**-x** *option=value*] [**-X** *option_file*]

Remarks

- This command supports operation on remote systems. See **Remote Operation** below.
- For an overview of all SD commands, see the *sd*(5) man page by typing **man 5 sd** on the command line.

DESCRIPTION

The roles of UNIX target and source systems require two processes known as the **daemon** and **agent**. For most purposes, the distinction between these two processes is invisible to the user and they can be viewed as a single process.

Each SD command interacts with the daemon and agent to perform its requested tasks.

The **swagentd** daemon process must be scheduled before a UNIX system is available as a target or source system. This can be done either manually or in the system start-up script. The **swagent** agent process is executed by **swagentd** to perform specific software management tasks. The **swagent** agent is never invoked by the user.

Remote Operation

You can enable SD to manage software on remote systems. To let the root user from a central SD *controller* (also called the *central management server* or *manager node*) perform operations on a remote *target* (also called the *host* or *agent*):

1) Install a special HP ServiceControl Manager fileset on the remote systems. This enables remote operations by automatically setting up the root, host, and template Access Control Lists (ACLs) on the remote machines and permitting root access from the controller system. To install the fileset, run the following command on each remote system:

swinstall -s *controller*:**/var/opt/mx/depot11 AgentConfig.SD-CONFIG**

NOTES:

- *controller* is the name of the central management server.
- If the target is running HP-UX 10.20, use the same command but substitute **depot10** for **depot11**.
- Targets previously set up by SD/OV to be managed by this controller do not need this step.
- SD does not require any other ServiceControl Manager filesets.

2) (Optional) **swinstall**, **swcopy**, and **swremove** have enhanced GUI interfaces for remote operations. Enable the enhanced GUIs by creating the **.sdkey** file on the controller. Use this command:

```
touch /var/adm/sw/.sdkey
```

NOTES:

- This step is not required when you use SD from within the HP ServiceControl Manager.
- See *sd*(5), *swinstall*(1M), *swcopy*(1M), *swjob*(1M), *swlist*(1M) or *swremove*(1M) for more information on interactive operations.

NOTE: You can also set up remote access by using **swacl** directly on the remote machines to grant root or non-root access to users from the controller system.

Options

The **swagentd** command supports the following options to control its behavior. (These options do not apply to **swagent**, which you cannot start from the command line.)

-k The *kill* option stops the currently running daemon. Stopping the daemon will not stop any agent processes currently performing management tasks (such as installing or removing software), but will cause any subsequent management requests to this host to be refused. This option is equivalent to sending a SIGTERM to the daemon that is running.

-n The *no fork* option runs the daemon as a synchronous process rather than the default behavior of forking to run it asynchronously. This is intended for running the daemon from other utilities that schedule processes, such as **init**.

-r The *restart* option stops the currently running daemon and restarts a new daemon. Because the **swagentd** daemon processes options only at startup, you must restart the daemon after you have modified any daemon options. Otherwise, the modified options have no effect.

-x *option=value*
Set the *option* to *value* and override the default value (or a value in an *option_file* specified with the **-X** option). Multiple **-x** options can be specified.

-X *option_file* Read the session options and behaviors from *options_file*.

EXTERNAL INFLUENCES

Default Options

In addition to the standard options, you can change SD behaviors and policy options by editing the system-wide default values found in the **/var/adm/sw/defaults** file. (Note that the user-specific default values in **$HOME/.swdefaults** do not apply to the agent or daemon.)

To specify values in the defaults file, you must use the following:

[*command_name*.]*option*=*value*

The optional *command_name* prefix denotes one of the SD commands. Using the prefix limits the value change to that command. If you leave the prefix off, the change applies to all commands that use the option.

You can also override **swagentd** default values from the command line with the **-x** or **-X** options:

command **-x** *option*=*value*

command **-X** *option_file*

NOTE: the only way to change default values for the agent is to modify the system-wide defaults file. You cannot change agent defaults from the command line.

The following section lists all of the keywords supported by the **swagentd** command. If a default value exists, it is listed after the "=".

Daemon Options

These options apply only to the daemon, **swagentd**. After changing daemon options, you must restart the daemon for these options to take effect (see the **-r** command-line option above).

agent=/usr/lbin/swagent
The location of the agent program invoked by the daemon.

logfile=/var/adm/sw/swagentd.log
This is the default log file for the **swagentd** daemon.

max_agents=-1
The maximum number of agents that are permitted to run simultaneously. The value of -1 means that there is no limit.

minimum_job_polling_interval=1
Defines in minutes how often the daemon wakes up to scan the job queue to determine if any scheduled jobs must be started. When set to 0, no scheduled jobs will be initiated.

rpc_binding_info=ncacn_ip_tcp:[2121] ncadg_ip_udp:[2121]
Defines the protocol sequence(s) and endpoint(s) on which the daemon listens and which the other commands use to contact the daemon. If the connection fails for one protocol sequence, the next is attempted. SD supports both the tcp (ncacn_ip_tcp:[2121]) and udp (ncadg_ip_udp:[2121]) protocol sequence on most platforms.

Agent Options

These options apply only to the agent, **swagent**. You cannot set these options directly from the command line. To set agent options, you must edit the system-wide defaults file. See the **Default Options** heading above for instructions.

alternate_source=
: If the **swinstall** or **swcopy** controller has set **use_alternate_source=true**, the target agent will consult and use the configured value of its own **alternate_source** option to determine the source that it will use in the install or copy.

  The agent's value for **alternate_source** is specified using the **host:path** syntax. If the host portion is not specified, the local host is used. If the path portion is not specified, the path sent by the command is used. If there is no configured value at all for **alternate_source**, the agent will apply the controller-supplied path to its own local host.

compress_cmd=/usr/contrib/bin/gzip
: Defines the command called by the source agent to compress files before transmission. If the **compression_type** is set to other than **gzip** or **compress**, this path must be changed.

compression_type=gzip
: Defines the default **compression_type** used by the agent when it compresses files during or after transmission. If **uncompress_files** is set to false, the **compression_type** is recorded for each file compressed so that the correct uncompression can later be applied during a **swinstall**, or a **swcopy** with **uncompress_files** set to true. The **compress_cmd** specified must produce files with the **compression_type** specified. The **uncompress_cmd** must be able to process files of the **compression_type** specified unless the format is **gzip**, which is uncompressed by the internal uncompressor (**funzip**). The only supported compression types are **compress** and **gzip**.

config_cleanup_cmd=/usr/lbin/sw/config_clean
: Defines the script called by the agent to perform release-specific configure cleanup steps.

install_cleanup_cmd=/usr/lbin/sw/install_clean
: Defines the script called by the agent to perform release-specific install cleanup steps immediately after the last postinstall script has been run. For an OS update, this script should at least remove commands that were saved by the **install_setup** script. This script is executed after all filesets have been installed, just before the reboot to the new operating system.

install_setup_cmd=/usr/lbin/sw/install_setup
: Defines the script called by the agent to perform release-specific install preparation. For an OS update, this script should at least copy commands needed for the checkinstall, preinstall, and postinstall scripts to a path where they can be accessed while the real commands are being updated. This script is executed before any kernel filesets are loaded.

kernel_build_cmd=/usr/sbin/mk_kernel
: Defines the script called by the agent for kernel building after kernel filesets have been loaded.

kernel_path=/stand/vmunix
: Defines the path to the system's bootable kernel. This path is passed to the **kernel_build_cmd** via the **SW_KERNEL_PATH** environment variable.

mount_cmd=/sbin/mount
: Defines the command called by the agent to mount all file systems.

reboot_cmd=/sbin/reboot
: Defines the command called by the agent to reboot the system after all filesets have been loaded, if any of the filesets required reboot.

remove_setup_cmd=/usr/lbin/sw/remove_setup
: Defines the script called by the agent to perform release-specific remove preparation. For an OS update, this script will invoke the **tlink** command when a fileset is removed.

rpc_binding_info_alt_source=ncadg_ip_udp:[2121]
Defines the protocol sequence(s) and endpoint(s) used when the agent attempts to contact an alternate source depot specified by the **alternate_source** option. SD supports both the **udp**(**ncadg_ip_udp:[2121]**) and **tcp**(**ncacn_ip_tcp:[2121]**) protocol sequence/endpoint.

source_depot_audit=true
If both source and target machine are updated to SD revision B.11.00 or later, the system administrator at the source depot machine can set this option to track *which* user pulls *which* software from a depot on the source machine and *when* the software is pulled. (A user running **swinstall/swcopy** from a target machine cannot set this option; only the administrator of the source depot machine can set it.)

When **swagent.source_depot_audit** is set to **true**, a **swaudit.log** file is created on the source depot (for writable directory depots) or in **/var/tmp** (for *tar* images, CD-ROMs, or other nonwritable depots).

Users can invoke the **swlist** interactive user interface (using **swlist -i -d**) to view, print, or save the audit information on a remote or local depot. Users can view audit information based on language preference, as long as the system has the corresponding SD message catalog files on it. For example, a user can view the source audit information in Japanese during one invocation of **swlist**, then view the same information in English at the next invocation.

system_file_path=/stand/system
Defines the path to the kernel's template file. This path is passed to the **system_prep_cmd** via the **SW_SYSTEM_FILE_PATH** environment variable.

system_prep_cmd=/usr/lbin/sysadm/system_prep
Defines the kernel build preparation script called by the agent. This script must do any necessary preparation so that control scripts can correctly configure the kernel about to be built. This script is called before any kernel filesets have been loaded.

uncompress_cmd=
Defines the command called by the target agent to uncompress files after transmission. This command processes files which were stored on the media in a compressed format. If the *compression_type* stored with the file is **gzip**, the internal uncompression (**funzip**) is used instead of the external **uncompress_cmd**. The default value for HP-UX is undefined.

Session File

swagentd and **swagent** do not use a session file.

Environment Variables

The environment variable that affects the **swagentd** and **swagent** commands is:

LANG Determines the language in which messages are displayed. If LANG is not specified or is set to the empty string, a default value of **C** is used. See the *lang*(5) man page by typing **man 5 sd** for more information.

Note: The language in which the SD agent and daemon log messages are displayed is set by the system configuration variable script, **/etc/rc.config.d/LANG**. For example, **/etc/rc.config.d/LANG**, must be set to **LANG=ja_JP.SJIS** or **LANG=ja_JP.eucJP** to make the agent and daemon log messages display in Japanese.

LC_ALL Determines the locale to be used to override any values for locale categories specified by the settings of **LANG** or any environment variables beginning with **LC_**.

LC_CTYPE
Determines the interpretation of sequences of bytes of text data as characters (e.g., single- versus multibyte characters in values for vendor-defined attributes).

LC_MESSAGES
Determines the language in which messages should be written.

LC_TIME
Determines the format of dates (*create_date* and *mod_date*) when displayed by **swlist**. Used by all utilities when displaying dates and times in **stdout**, **stderr**, and **logging**.

TZ Determines the time zone for use when displaying dates and times.

Signals

The daemon ignores SIGHUP, SIGINT and SIGQUIT. It immediately exits gracefully after receiving SIGTERM and SIGUSR2. After receiving SIGUSR1, it waits for completion of a copy or remove from a depot session before exiting, so that it can register or unregister depots. Requests to start new sessions are refused during this wait.

The agent ignores SIGHUP, SIGINT, and SIGQUIT. It immediately exits gracefully after receiving SIGTERM, SIGUSR1, or SIGUSR2. Killing the agent may leave corrupt software on the system, and thus should only be done if absolutely necessary. Note that when an SD command is killed, the agent does not terminate until completing the task in progress.

Locking

The **swagentd** ensures that only one copy of itself is running on the system.

Each copy of **swagent** that is invoked uses appropriate access control for the operation it is performing and the object it is operating on.

RETURN VALUES

When the **-n** option is not specified, the **swagentd** returns:

0	When the daemon is successfully initialized and is now running in the background.
non-zero	When initialization failed and the daemon terminated.

When the **-n** option is specified, the **swagentd** returns:

0	When the daemon successfully initialized and then successfully shutdown.
non-zero	When initialization failed or the daemon unsuccessfully terminated.

DIAGNOSTICS

The **swagentd** and **swagent** commands log events to their specific logfiles.

The **swagent** (target) log files cannot be relocated. They always exist relative to the root or depot target path (e.g. **/var/adm/sw/swagent.log** for the root **/** and **/var/spool/sw/swagent.log** for the depot **/var/spool/sw**).

You can view the target log files using the **swjob** or **sd** command.

Daemon Log

The daemon logs all events to **/var/adm/sw/swagentd.log**. (The user can specify a different logfile by modifying the **logfile** option.)

Agent Log

When operating on (alternate) root file systems, the **swagent** logs messages to the file **var/adm/sw/swagent.log** beneath the root directory (e.g. **/** or an alternate root directory).

Source Depot Audit Log

If both source and target machine are updated to HP-UX version 10.30 or later, the system administrator at the source depot machine can track *which* user pulls *which* software from a depot on the source machine and *when* the software is pulled. Refer to the **swagent.source_depot_audit** option for more information.

When operating on software depots, the **swagent** logs messages to the file **swagent.log** beneath the depot directory (e.g. **/var/spool/sw**). When accessing a read-only software depot (e.g. as a source), the **swagent** logs messages to the file **/tmp/swagent.log**.

EXAMPLES

To start the daemon:

```
/usr/sbin/swagentd
```

To restart the daemon:

```
/usr/sbin/swagentd -r
```

To stop the daemon:

```
/usr/sbin/swagentd -k
```

S

FILES

`/usr/lib/sw/sys.defaults`
Contains the master list of current SD options (with their default values).

`/var/adm/sw/`
The directory which contains all configurable and non-configurable data for SD. This directory is also the default location of logfiles.

`/var/adm/sw/defaults`
Contains the active system-wide default values for some or all SD options.

`/var/adm/sw/host_object`
The file which stores the list of depots registered at the local host.

AUTHOR

`swagentd` was developed by the Hewlett-Packard Company. **`swagent`** was developed by the Hewlett-Packard Company and Mark H. Colburn (see *pax*(1)).

SEE ALSO

sd(5), sd(4), swpackage(4), swacl(1M), swask(1M), swconfig(1M), swcopy(1M), swinstall(1M), swjob(1M), swlist(1M), swmodify(1M), swpackage(1M), swreg(1M), swremove(1M), swverify(1M), install-sd(1M).

Software Distributor Administration Guide, available at **`http://docs.hp.com`**.

SD customer web site at **`http://software.hp.com/SD_AT_HP/`**.

NAME

swapinfo - system paging space information

SYNOPSIS

/usr/sbin/swapinfo [**-mtadfnrMqw**]

DESCRIPTION

swapinfo prints information about device and file system paging space. (Note: the term ‘swap’ refers to an obsolete implementation of virtual memory; HP-UX actually implements virtual memory by way of paging rather than swapping. This command and others retain names derived from ‘swap’ for historical reasons.)

By default, **swapinfo** prints to standard output a two line header as shown here, followed by one line per paging area:

```
                Kb        Kb        Kb        PCT       START/    Kb
    TYPE        AVAIL     USED      FREE      USED      LIMIT     RESERVE  PRI       NAME
```

The fields are:

TYPE One of:

- **dev** Paging space residing on a mass storage device, either taking up the entire device or, if the device contains a file system, taking up the space between the end of the file system and the end of the device. This space is exclusively reserved for paging, and even if it is not being used for paging, it cannot be used for any other purpose. Device paging areas typically provide the fastest paging.
- **fs** Dynamic paging space available from a file system. When this space is needed, the system creates files in the file system and uses them as paging space. File system paging is typically slower than device paging, but allows the space to be used for other things (user files) when not needed for paging.
- **localfs** File system paging space (see **fs** above) on a file system residing on a local disk.
- **network** File system paging space (see **fs** above) on a file system residing on another machine. This file system would have been mounted on the local machine via NFS.
- **reserve** Paging space on reserve. This is the amount of paging space that could be needed by processes that are currently running, but that has not yet been allocated from one of the above paging areas. See "Paging Allocation" below.
- **memory** Memory paging area (also known as pseudo-swap). This is the amount of system memory that can be used to hold pages in the event that all of the above paging areas are used up. See "Paging Allocation" below. This line appears only if memory paging is enabled.

S

Kb AVAIL The total available space from the paging area, in blocks of 1024 bytes (rounded to nearest whole block if necessary), including any paging space already in use.

For file system paging areas the value is not necessarily constant. It is the current space allocated for paging (even if not currently used), plus the free blocks available on the file system to ordinary users, minus RESERVE (but never less than zero). AVAIL is never more than LIMIT if LIMIT is non-zero. Since paging space is allocated in large chunks, AVAIL is rounded down to the nearest full allocation chunk.

For the memory paging area this value is also not necessarily constant, because it reflects allocation of memory by the kernel as well as by processes that might need to be paged.

Kb USED The current number of 1-Kbyte blocks used for paging in the paging area. For the memory paging area, this count also includes memory used for other purposes and thus unavailable for paging.

Kb FREE The amount of space that can be used for future paging. Usually this is the difference between Kb AVAIL and Kb USED. There could be a difference if some portion of a device paging area is unusable, perhaps because the size of the paging area is not a multiple of the allocation chunk size, or because the tunable parameter **maxswapchunks** is not set high enough.

PCT USED The percentage of capacity in use, based on Kb USED divided by Kb AVAIL; 100% if Kb AVAIL is zero.

START/LIMIT

For device paging areas, START is the block address on the mass storage device of the start of the paging area. The value is normally 0 for devices dedicated to paging, or the end of the file system for devices containing both a file system and paging space.

For file system paging areas, LIMIT is the maximum number of 1-Kbyte blocks that will be used for paging, the same as the *limit* value given to **swapon**. A file system LIMIT value of **none** means there is no fixed limit; all space is available except that used for files, less the blocks represented by **minfree** (see *fs*(4)) plus RESERVE.

RESERVE For device paging areas, this value is always "—". For file system paging areas, this value is the number of 1-Kbyte blocks reserved for file system use by ordinary users, the same as the *reserve* value given to **swapon**.

PRI The same as the *priority* value given to **swapon**. This value indicates the order in which space is taken from the devices and file systems used for paging. Space is taken from areas with lower priority values first. *priority* can have a value between 0 and 10. See "Paging Allocation" below.

NAME For device paging areas, the block special file name whose major and minor numbers match the device's ID. The **swapinfo** command searches the **/dev** tree to find device names. If no matching block special file is found, **swapinfo** prints the device ID (major and minor values), for example, **28,0x15000**.

For file system swap areas, NAME is the name of a directory on the file system in which the paging files are stored.

Paging Allocation

Paging areas are enabled at boot time (for device paging areas configured into the kernel) or by the **swapon** command (see *swapon*(1M)), often invoked by **/sbin/init.d/swap_start** during system initialization based on the contents of **/etc/fstab**. When a paging area is enabled, some portion of that area is allocated for paging space. For device paging areas, the entire device is allocated, less any leftover fraction of an allocation chunk. (The size of an allocation chunk is controlled by the tunable parameter **swchunk**, and is typically 2 MB.) For file system paging areas, the *minimum* value given to **swapon** (rounded up to the nearest allocation chunk) is allocated.

When a process is created, or requests additional space, space is reserved for it by increasing the space shown on the **reserve** line above. When paging activity actually occurs, space is used in one of the paging areas (the one with the lowest priority number that has free space available, already allocated), and that space will be shown as used in that area.

The sum of the space used in all of the paging areas, plus the amount of space reserved, can never exceed the total amount allocated in all of the paging areas. If a request for more memory occurs which would cause this to happen, the system tries several options:

1. The system tries to increase the total space available by allocating more space in file system paging areas.
2. If all file system paging areas are completely allocated and the request is still not satisfied, the system will try to use memory paging as described on the **memory** line above. (Memory paging is controlled by the tunable parameter **swapmem_on**, which defaults to 1 (on). If this parameter is turned off, the **memory** line will not appear.)
3. If memory paging also cannot satisfy the request, because it is full or turned off, the request is denied.

Several implications of this procedure are noteworthy for understanding the output of **swapinfo**:

- Paging space will not be allocated in a file system paging area (except for the *minimum* specified when the area is first enabled) until all device paging space has been reserved, even if the file system paging area has a lower priority value.
- When paging space is allocated to a file system paging area, that space becomes unavailable for user files, even if there is no paging activity to it.
- Requests for more paging space will fail when they cannot be satisfied by reserving device, file system, or memory paging, even if some of the reserved paging space is not yet in use. Thus it is possible for requests for more paging space to be denied when some, or even all, of the paging areas show zero usage — space in those areas is completely reserved.

- System available memory is shared between the paging subsystem and kernel memory allocators. Thus, the system may show memory paging usage before all available disk paging space is completely reserved or fully allocated.

Logical Volume Manager (LVM)

The **swapinfo** command displays swap logical volume if the system was installed with LVM. To modify swap logical volume, refer to the LVM commands and manpages for **lvlnboot** and **lvrmboot**. For example, to remove a swap logical volume, run the following LVM command:

```
lvrmboot -s
```

Options

swapinfo recognizes the following options:

-m Display the AVAIL, USED, FREE, LIMIT, and RESERVE values in Mbytes instead of Kbytes, rounding off to the nearest whole Mbyte (multiples of 1024^2). The output header format changes from **Kb** to **Mb** accordingly.

-t Add a totals line with a TYPE of **total**. This line totals only the paging information displayed above it, not all paging areas; this line might be misleading if a subset of **-dfrM** is specified.

-a Show all device paging areas, including those configured into the kernel but currently disabled. (These are normally omitted.) The word **disabled** appears after the NAME, and the Kb AVAIL, Kb USED, and Kb FREE values are 0. The **-a** option is ignored unless the **-d** option is present or is true by default.

-d Print information about device paging areas only. This modifies the output header appropriately.

-f Print information about file system paging areas only. This modifies the output header appropriately.

-n Categorize file system paging area information into **localfs** areas and **network** areas, instead of calling them both **fs** areas.

-r Print information about reserved paging space only.

-M Print information about memory paging space only.

The **-d**, **-f**, **-n**, **-r** and **-M** options can be combined. The default is **-dfnrM**.

-q Quiet mode. Print only a total "Kb AVAIL" value (with the **-m** option, Mb AVAIL); that is, the total paging space available on the system (device, file system, reserve, or memory paging space only if **-d**, **-f**, **-r**, or **-M** is specified), for possible use by programs that want a quick total. If **-q** is specified, the **-t** and **-a** options are ignored.

-w Print a warning about each device paging area that contains wasted space; that is, any device paging area whose allocated size is less than its total size. This option is effective only if **-d** is also specified or true by default.

RETURN VALUE

swapinfo returns 0 if it completes successfully (including if any warnings are issued), or 1 if it reports any errors.

DIAGNOSTICS

swapinfo prints messages to standard error if it has any problems.

EXAMPLES

List all file system paging areas with a totals line:

```
swapinfo -ft
```

WARNINGS

swapinfo needs kernel access for some information. If the user does not have appropriate privileges for kernel access, **swapinfo** will print a warning and assume that the defaults for that information have not been changed.

Users of **swapinfo** must not rely on the exact field widths and spacing of its output, as these will vary depending on the system, the release of HP-UX, and the data to be displayed.

The information in this manual page about paging allocation and other implementation details may change without warning; users should not rely on the accuracy of this information.

AUTHOR

swapinfo was developed by HP.

SEE ALSO

lvlnboot(1M), lvrmboot(1M), swapon(1M), swapon(2), fstab(4), fs(4).

NAME

swapon - enable device or file system for paging

SYNOPSIS

Preferred Forms:

/usr/sbin/swapon -a [**-u**] [**-t** *type*]...

/usr/sbin/swapon [**-e** | **-f**] [**-p** *priority*] [**-u**] *device* ...

/usr/sbin/swapon [**-m** *min*] [**-l** *limit*] [**-r** *reserve*] [**-p** *priority*] *directory* ...

Obsolescent Form:

/usr/sbin/swapon *directory* [*min limit reserve priority*]

DESCRIPTION

The **swapon** command enables devices or file systems on which paging is to take place. (**NOTE:** the term 'swap' refers to an obsolete implementation of virtual memory; HP-UX actually implements virtual memory by way of paging rather than swapping. This command and others retain names derived from 'swap' for historical reasons.)

By enabling a **device** for paging, the device can be accessed directly (without going through the file system) during paging activity. When a **file system** is enabled for paging, the device(s) on which the file system resides are accessed indirectly through the file system. There are advantages and disadvantages to both type of paging. Keep the following tradeoffs in mind when enabling devices or file systems for paging.

Paging directly to a **device** is significantly faster than doing so through the file system. However, the space on the device that is allocated to paging cannot be used for anything else, even if it is not being actively used for paging.

Paging through a **file system**, while slower, provides a more efficient use of the space on the device. Space that is not being used for paging in this case can be used by the file system. Paging across a network to a remote machine is always file system paging.

The system begins by paging on only a single device so that only one disk is required at bootstrap time. Calls to **swapon** normally occur in the system startup script **/sbin/init.d/swap_start** making all paging space available so that the paging activity is interleaved across several disks.

Normally, the **-a** argument is given, causing all devices marked as **swap** and all file systems marked as **swapfs** in the file **/etc/fstab** to be made available to the paging system. By using the fields in **/etc/fstab** (*special_file_name* or *directory*; see *fstab*(5)), the system determines which block device or file system to use. The *special_file_name* specified for each **swap** entry must specify a block special file. The *directory* specified for each **swapfs** entry must specify a directory within the file system to be enabled.

S

The second form of **swapon** enables individual block devices to be used for paging. The *device* name must specify a block special file. If more than one device is given, any options specified will be applied to all devices. If a file system exists on the specified block device and neither an **-e** nor **-f** option is specified, **swapon** fails and an error message is given. This prevents a file system from being inadvertently destroyed. To request paging in the space between the end of the file system and the end of the device, use **-e**. To force paging to a device containing a file system (destroying the file system), the **-f** option can be used. Use this with extreme caution!

In either of the previous forms, an attempt to enable paging to a device will fail and a warning message will be issued if **swapon** determines that the device is being used by the **savecore** command to retrieve system dump information (see *savecore*(1M)). The **-u** option can be used to forcibly enable paging to devices being used by **savecore**; however, this may overwrite system dump information contained on the device.

The last two forms of **swapon** provide two different methods for enabling file systems for paging. The third form is the preferred method, with the fourth being provided only for backward compatibility. The *directory* name specifies a directory on the file system that is to be enabled for paging. A directory named **/paging** is created at the root of the specified file system (unless the file system's name ends with **/paging**). All paging files are created within this directory. The optional arguments to the fourth form have the same meaning as the arguments to the options in the third form. Note that, in the fourth form, if any of the optional arguments are specified, all must be specified. In the third form, if more than one directory is given, any options specified will be applied to all directories.

After a file system has been enabled for paging, the optional arguments can be modified by subsequent **swapon** commands.

Options

swapon recognizes the following options and arguments:

-a Cause all devices marked as **swap** and all file systems marked as **swapfs** in the file **/etc/fstab** to be made available to the paging system. The *options* field in **/etc/fstab** entries is read by **swapon**, and must contain elements formatted as follows:

min=*min*	See the **-m** option for the value of *min*.
lim=*limit*	See the **-l** option for the value of *limit*. (File system paging areas only.)
res=*reserve*	See the **-r** option for the value of *reserve*. (File system paging areas only.)
pri=*priority*	See the **-p** option for the value of *priority*. (File system paging areas only.)
end	See the **-e** option for the meaning of this option. (Device paging areas only.)

See *fstab*(4) for an example entry.

-e Use space after the end of the file system on the block device for paging. An error message is returned if no file system is found on the device. This option cannot be used with the **-f** option. Do not confuse this with paging to a file system. This option is for use with a disk that has *both* a file system and dedicated paging space on it.

-f Force the *device* to be enabled, which will destroy the file system on it. Use with extreme caution. Normally, if a file system exists on the *device* to be enabled, **swapon** fails and displays an error message. This option cannot be used with the **-e** option.

-l *limit* *limit* specifies the maximum space the paging system is allowed to take from the disk, provided space is available that is not reserved for exclusive use by the file system. The value of *limit* is rounded up so that it is a multiple of the paging allocation chunk size, which is set with the kernel tunable parameter **swchunk** (see *config*(1M) and *swapinfo*(1M)). See *WARNINGS*. The default value for *limit* is 0, indicating there is no limit to the amount of file system space the paging system can use.

limit can be specified in decimal (no prefix), octal (**0** prefix), or hexadecimal (**0x** prefix). It may be specified in units of kilobytes (**k** suffix), megabytes (**M** suffix), or file system blocks (no suffix). (A kilobyte is 1024 bytes; a megabyte is 1024 kilobytes; the size of a file system block is determined by the administrator when the file system is created.)

-m *min* *min* indicates the space the paging system will initially take from the file system. The value of *min* is rounded up so that it is a multiple of the paging allocation chunk size, which is set with the kernel tunable parameter **swchunk** (see *config*(1M) and *swapinfo*(1M)). The default value for *min* is 0, indicating no paging space is to be allocated initially. *min* can be specified in the same forms as *limit*, above.

-p *priority* *priority* indicates the order in which space is taken from the file systems and devices used for paging. Space is taken from the systems with lower priority numbers first. Under most circumstances, space is taken from device paging areas before file system paging areas, regardless of priority. See "Paging Allocation" in *swapinfo*(1M) for more information. *priority* can have a value from 0 to 10 and has a default value of 1.

-r *reserve* *reserve* specifies the space, in addition to the space currently occupied by the file system, that is reserved for file system use only, making it unavailable to the paging system. This reserved space is in addition to the minimum free space specified by the administrator when the file system was created. See *WARNINGS*. The default value for *reserve* is 0 indicating that no file system space is reserved for file system use only. *reserve* can be specified in the same forms as *limit*, above.

-t *type* Restrict the type of the paging area. If the **-t** option is omitted, all of the paging areas defined in **/etc/fstab** are made available. *type* can have one of the following values:

`dev`	Device paging areas.
`fs`	File system paging areas.
`local`	Paging areas defined on the local system.
`remote`	Paging areas defined on remote systems.

`-u` Unlock block device files which are being used by the **`savecore`** command. Normally, **`swapon`** will not enable paging on a device if it is being used by **`savecore`** to retrieve system dump information. The list of devices in use is maintained in the file **`/etc/savecore.LCK`**. This option forces the device to be enabled, which may overwrite any system dump information contained on the device. This option should be used with extreme caution.

RETURN VALUE

`swapon` returns one of the following values:

`0`	Successful completion.
`>0`	An error condition occurred.

EXAMPLES

The first two examples enable paging to the file system containing the **`/paging`** directory. The maximum number of file system blocks available to the paging system is set to 5000, the number of file system blocks reserved for file system use only is set to 10000, and the priority is set to 2. The number of file system blocks initially taken by the paging system defaults to 0 in the first example, and is set to 0 in the second example. On a file system with the default 8kB block size, these examples allocate approximately 40MB of file system paging.

```
/usr/sbin/swapon -l 5000 -r 10000 -p 2 /paging
/usr/sbin/swapon /paging 0 5000 10000 2
```

This example enables paging to two block devices and sets the priority of both devices to 0.

```
/usr/sbin/swapon -p 0 /dev/dsk/c10t0d0 /dev/dsk/c13t0d0
```

This example enables paging to a block device, using the space after the end of the file system for paging and letting the priority default to 1.

```
/usr/sbin/swapon -e /dev/dsk/c4t0d0
```

This example enables paging to a block device, forcing paging even if a file system exists on the device.

```
/usr/sbin/swapon -f /dev/dsk/c12t0d0
```

WARNINGS

Once file system blocks have been allocated for paging space, the file system cannot be unmounted unless the system is rebooted.

S

If any paging area becomes unavailable while the system is running, for example if a network failure occurs while paging to a remote system, the system will immediately halt.

The file system block size used by the **`-l`**, **`-m`**, and **`-r`** options varies between file systems, and is defined by the system administrator at the time the file system is created. The **`dumpfs`** command can be used to determine the block size for a particular file system (see *dumpfs*(1M)).

When using the **`-l`** and **`-r`** options, the reserve space specified by the **`-r`** option takes precedence over the **`-l`** option. Thus, if:

D	= Total disk space available to ordinary users
R	= Reserve space specified by the **`-r`** option
limit	= Paging space limit specified by the **`-l`** option
L	= Space currently available to the paging system
F	= Space currently occupied by the file system

the following relationships hold:

$F + R + limit < D$	In normal operation
$L = 0$	If $F + R >= D$

$0 <= L <= limit$	If $F + R + limit >= D$

FILES

/dev/dsk/c*card***t***target***d***device*	Normal paging devices
/etc/fstab	File system table
/etc/savecore.LCK	List of devices being used by **savecore**

AUTHOR

swapon was developed by HP and the University of California, Berkeley.

SEE ALSO

config(1M), savecore(1M), swapinfo(1M), swapon(2), fstab(4).

NAME

swask - ask for user response

SYNOPSIS

swask [**-v**] [**-c** *catalog*] [**-C** *session_file*] [**-f** *software_file*] [**-J** *jobid*] [**-Q** *date*] [**-s** *source*] [**-S** *session_file*] [**-t** *target_file*] [**-x** *option=value*] [**-X** *options_file*] [*software_selections*] [**@** *target_selections*]

Remarks

- This command supports operation on remote systems. See **Remote Operation** below.
- For an overview of all SD commands, see the *sd*(5) man page by typing **man 5 sd** on the command line.

DESCRIPTION

The **swask** command runs interactive software **request** scripts for the software objects selected to one or more targets specified by *target_selections*. These scripts store the responses in a **response** file (named **response**) for later use by the **swinstall** and **swconfig** commands. The **swinstall** and **swconfig** commands can also run the interactive request scripts directly, using the **ask** option.

If the **-s** option is specified, software is selected from the distribution source. If the **-s** option is not specified, software installed on the target systems is selected. For each selected software that has a request script, executing that script generates a response file. By specifying the **-c** *catalog* option, **swask** stores a copy of the response file to that catalog for later use by **swinstall** or **swconfig**.

Remote Operation

You can enable SD to manage software on remote systems. To let the root user from a central SD *controller* (also called the *central management server* or IR "manager node") perform operations on a remote *target* (also called the *host* or *agent*):

1) Install a special HP ServiceControl Manager fileset on the remote systems. This enables remote operations by automatically setting up the root, host, and template Access Control Lists (ACLs) on the remote machines and permitting root access from the controller system. To install the fileset, run the following command on each remote system:

swinstall -s *controller* **:/var/opt/mx/depot11 AgentConfig.SD-CONFIG**

NOTES:

- *controller* is the name of the central management server.
- If the target is running HP-UX 10.20, use the same command but substitute **depot10** for **depot11**.
- Targets previously set up by SD/OV to be managed by this controller do not need this step.
- SD does not require any other ServiceControl Manager filesets.

2) (Optional) **swinstall**, **swcopy**, and **swremove** have enhanced GUI interfaces for remote operations. Enable the enhanced GUIs by creating the **.sdkey** file on the controller. Use this command:

touch /var/adm/sw/.sdkey

NOTES:

- This step is not required when you use SD from within the HP ServiceControl Manager.
- See *sd(5)*, *swinstall(1M)*, *swcopy(1M)*, *swjob(1M)*, *swlist(1M)*or *swremove(1M)* for more information on interactive operations.

NOTE: You can also set up remote access by using **swacl** directly on the remote machines to grant root or non-root access to users from the controller system.

S

Options

The **swask** command supports the following options:

-v	Turns on verbose output to stdout.
-c *catalog*	Specifies the pathname of an exported catalog which stores the response files created by the request script. **swask** creates the catalog if it does not already exist.

If the -c *catalog* option is omitted and the source is local, **swask** copies the response files into the source depot, <*distribution.path*>**/***catalog*.

-C *session_file*
Saves the current options and operands to *session_file*. You can enter a relative or absolute path with the file name. The default directory for session files is **$HOME/.sw/sessions/**. You can recall a session file with the **-S** option.

-f *software_file*
Reads the list of *software_selections* from *software_file* instead of (or in addition to) the command line.

-s *source* Specifies the source depot (or tape) from which software is selected for the ask operation. (SD can read both **tar** and **cpio** tape depots.)

-S *session_file*
Executes **swask** based on the options and operands saved from a previous session, as defined in *session_file*. You can save session information from a command-line session with the **-C** *session_file* option.

-t *targetfile* Specifies a default set of *targets* for **swask.**

-x *option=value*
Sets the session *option* to *value* and overrides the default value (or a value in an alternate *option_file* specified with the **-X** option). Multiple **-x** options can be specified.

-X *option_file* Reads the session options and behaviors from *option_file*.

Operands

swask supports two types of operands: *software selections* followed by *target selections*. These operands are separated by the "@" (at) character. This syntax implies that the command operates on "software selections at targets".

Software Selections

The *selections* operands consist of *software_selections*.

swask supports the following syntax for each *software_selection*:

bundle[**.***product*[**.***subproduct*][**.***fileset*]][**,***version*]

product[**.***subproduct*][**.***fileset*][**,***version*]

- The **=** (equals) relational operator lets you specify selections with the following shell wildcard and pattern-matching notations:

  [ ], *****, **?**

- *Bundles* and *subproducts* are recursive. *Bundles* can contain other *bundles* and *subproducts* can contain other *subproducts*.
- The **\*** software specification selects all products. Use this specification with caution.

The **version** component has the form:
[**,r** <*op*> *revision*][**,a** <*op*> *arch*][**,v** <*op*> *vendor*]
[**,c** <*op*> *category*][**,q=***qualifier*][**,l=***location*]
[**,fr** <*op*> *revision*][**,fa** <*op*> *arch*]

- *location* applies only to installed software and refers to software installed to a location other than the default product directory.
- **fr** and **fa** apply only to filesets.
- The <*op*> (relational operator) component can be of the form:

  =, **==**, **>=**, **<=**, **<**, **>**, or **!=**

  which performs individual comparisons on dot-separated fields. For example, **r>=B.10.00** chooses all revisions greater than or equal to **B.10.00**. The system compares each dot-separated field to find matches.
- The **=** (equals) relational operator lets you specify selections with the shell wildcard and pattern-matching notations:

S

[ ], *****, **?**, **!**

For example, the expression **r=1[01].*** returns any revision in version 10 or version 11.

- All version components are repeatable within a single specification (e.g. **r>=AA.12**, **r<AA.20**). If multiple components are used, the selection must match all components.
- Fully qualified software specs include the **r=**, **a=**, and **v=** version components even if they contain empty strings. For installed software, **l=**, is also included.
- No space or tab characters are allowed in a software selection.
- The software *instance_id* can take the place of the version component. It has the form:

  [*instance_id*]

  within the context of an exported catalog, where *instance_id* is an integer that distinguishes versions of products and bundles with the same tag.

Target Selections

swask supports the following syntax for each *target_selection*.

[*host*][**:**][**/***directory*]

The **:** (colon) is required if both a host and directory are specified.

EXTERNAL INFLUENCES

Default Options

In addition to the standard options, several SD behaviors and policy options can be changed by editing the default values found in:

/var/adm/sw/defaults	the system-wide default values.
$HOME/.swdefaults	the user-specific default values.

Values must be specified in the defaults file using this syntax:

[*command_name***.**]*option***=***value*

The optional *command_name* prefix denotes one of the SD commands. Using the prefix limits the change in the default value to that command. If you leave the prefix off, the change applies to all commands.

You can also override default values from the command line with the **-x** or **-X** options:

command **-x** *option***=***value*

command **-X** *option_file*

The following section lists all of the keywords supported by the **swask** commands. If a default value exists, it is listed after the "=".

admin_directory=/var/adm/sw (for normal mode)
admin_directory=/var/home/LOGNAME/sw (for nonprivileged mode)

The location for SD logfiles and the default parent directory for the installed software catalog. The default value is **/var/adm/sw** for normal SD operations. When SD operates in nonprivileged mode (that is, when the **run_as_superuser** default option is set to **true**):

- The default value is forced to **/var/home/LOGNAME/sw**.
- The path element **LOGNAME** is replaced with the name of the invoking user, which SD reads from the system password file.
- If you set the value of this option to **HOME/***path*, SD replaces **HOME** with the invoking user's home directory (from the system password file) and resolves *path* relative to that directory. For example, **HOME/my_admin** resolves to the **my_admin** directory in your home directory.
- If you set the value of the **installed_software_catalog** default option to a relative path, that path is resolved relative to the value of this option.

SD's nonprivileged mode is intended only for managing applications that are specially designed and packaged. This mode cannot be used to manage the HP-UX operating system or patches to it. For a full explanation of nonprivileged SD, see the *Software Distributor*

S

Administration Guide, available at the **http://docs.hp.com** web site.

See also the **installed_software_catalog** and **run_as_superuser** options.

ask=true
Executes the request script, if one is associated with the selected software, and stores the user response in a file named **response**.

If **ask=as_needed**, the **swask** command first determines if a response file already exists in the catalog and executes the request script only when a response file is absent.

autoselect_dependencies=true
Controls the automatic selection of prerequisite and corequisite software that is not explicitly selected by the user. When set to **true**, requisite software will be automatically selected for configuration. When set to **false**, requisite software which is not explicitly selected will not be automatically selected for configuration.

autoselect_patches=true
Automatically selects the latest patches (based on superseding and ancestor attributes) for a software object that a user selects. The **patch_filter** option can be used in conjunction with **autoselect_patches** to limit which patches will be selected. Requires patches that are in an enhanced SD format. Patches not in enhanced format will not respond to **autoselect_patches**.

enforce_scripts=true
Controls the handling of errors generated by scripts. If *true*, **swask** stops and an error message appears. The message gives the script location and says execution cannot proceed until the problem is fixed. If **false,** all script errors are treated as warnings, and **swask** attempts to continue operation. A message appears giving the script location and saying that execution will proceed.

installed_software_catalog=products
Defines the directory path where the Installed Products Database (IPD) is stored. This information describes installed software. When set to an absolute path, this option defines the location of the IPD. When this option contains a relative path, the SD controller appends the value to the value specified by the **admin_directory** option to determine the path to the IPD. For alternate roots, this path is resolved relative to the location of the alternate root. This option does not affect where software is installed, only the IPD location.

This option permits the simultaneous installation and removal of multiple software applications by multiple users or multiple processes, with each application or group of applications using a different IPD.

Caution: use a specific **installed_software_catalog** to manage a specific application. SD does not support multiple descriptions of the same application in multiple IPDs.

See also the **admin_directory** and **run_as_superuser** options, which control SD's nonprivileged mode. (This mode is intended only for managing applications that are specially designed and packaged. This mode cannot be used to manage the HP-UX operating system or patches to it. For a full explanation of nonprivileged SD, see the *Software Distributor Administration Guide*, available at the **http://docs.hp.com** web site.)

log_msgid=0
Controls the log level for the events logged to the command log file, the target agent log file, and the source agent log file by prepending identification numbers to log file messages:
0 No such identifiers are prepended (default).
1 Applies to ERROR messages only.
2 Applies to ERROR and WARNING messages.
3 Applies to ERROR, WARNING, and NOTE messages.
4 Applies to ERROR, WARNING, NOTE, and certain other log file messages.

logdetail=false
Controls the amount of detail written to the logfile. When set to **true**, this option adds detailed task information (such as options specified, progress statements, and additional summary information) to the logfile. This information is in addition to log information controlled by the **loglevel** option.

S

See **loglevel** below and the *sd*(5) manual page, by typing **man 5 sd**, for more information.

logfile=/var/adm/sw/swask.log
Defines the default log file for swask.

loglevel=1
Controls the log level for the events logged to the command logfile and the target agent logfile. A value of
0 provides no information to the logfile.
1 enables verbose logging of key events to the log files.
2 enables very verbose logging, including per-file messages, to the log files.

patch_filter=*.*
Used in conjunction with the **autoselect_patches** or **patch_match_target** options to filter the available patches to meet the criteria specified by the filter. A key use is to allow filtering by the "category" attribute. Requires patches that are in an enhanced SD patch format.

run_as_superuser=true
This option controls SD's nonprivileged mode. This option is ignored (treated as true) when the invoking user is super-user.

When set to the default value of true, SD operations are performed normally, with permissions for operations either granted to a local super-user or set by SD ACLs. (See *swacl*(1M) for details on ACLs.)

When set to false and the invoking user is local and is *not* super-user, nonprivileged mode is invoked:

- Permissions for operations are based on the user's file system permissions.
- SD ACLs are ignored.
- Files created by SD have the uid and gid of the invoking user, and the mode of created files is set according to the invoking user's umask.

SD's nonprivileged mode is intended only for managing applications that are specially designed and packaged. This mode cannot be used to manage the HP-UX operating system or patches to it. For a full explanation of nonprivileged SD, see the *Software Distributor Administration Guide*, available at the **http://docs.hp.com** web site.

See also the **admin_directory** and **installed_software_catalog** options.

verbose=1
Controls the verbosity of the output (stdout):
0 disables output to stdout. (Error and warning messages are always written to stderr).
1 enables verbose messaging to stdout.

S

Session Files

Each invocation of **swask** defines a task session. The invocation options, source information, software selections, and target hosts are saved before the task actually commences. This lets you re-execute the command even if the session ends before proper completion.

Each session is saved to the file **$HOME/.sw/sessions/swask.last**. This file is overwritten by each invocation of **swask**.

To save session information in a different location, execute **swask** with the **-C** *session__file* option.

A session file uses the same syntax as the defaults files. You can specify an absolute path for a session file. If you do not specify a directory, the default location for a session file is **$HOME/.sw/sessions/**.

To re-execute a session, specify the session file as the argument for the **-S** *session__file* option.

When you re-execute a session file, the values in the session file take precedence over values in the system defaults file. Likewise, any command line options or parameters that you specify when you invoke **swask** take precedence over the values in the session file.

Software and Target Lists

You can use files containing software and target selections as input to the **swask** command. See the **-f** and **-t** options for more information.

Environment Variables

The environment variable that affects the **swask** command is:

LANG Determines the language in which messages are displayed. If LANG is not specified or is set to the empty string, a default value of **C** is used. See *lang*(5) for more information.

NOTE: The language in which the SD agent and daemon log messages are displayed is set by the system configuration variable script, **/etc/rc.config.d/LANG**. For example, **/etc/rc.config.d/LANG**, must be set to **LANG=ja_JP.SJIS** or **LANG=ja_JP.eucJP** to make the agent and daemon log messages display in Japanese.

LC_ALL Determines the locale to be used to override any values for locale categories specified by the settings of **LANG** or any environment variables beginning with **LC_**.

LC_CTYPE
Determines the interpretation of sequences of bytes of text data as characters (e.g., single- versus multibyte characters in values for vendor-defined attributes).

LC_MESSAGES
Determines the language in which messages should be written.

LC_TIME
Determines the format of dates (*create_date* and *mod_date*) when displayed by **swlist**. Used by all utilities when displaying dates and times in **stdout**, **stderr**, and **logging**.

TZ Determines the time zone for use when displaying dates and times.

Environment variables that affect scripts:

SW_CATALOG
Holds the path to the Installed Products Database (IPD), relative to the path in the SW_ROOT_DIRECTORY environment variable. Note that you can specify a path for the IPD using the **installed_software_catalog** default option.

SW_CONTROL_DIRECTORY
Defines the current directory of the script being executed, either a temporary catalog directory, or a directory within in the Installed Products Database (IPD). This variable tells scripts where other control scripts for the software are located (e.g. subscripts).

SW_CONTROL_TAG
Holds the tag name of the control_file being executed. When packaging software, you can define a physical name and path for a control file in a depot. This lets you define the control_file with a name other than its tag and lets you use multiple control file definitions to point to the same file. A control_file can query the **SW_CONTROL_TAG** variable to determine which tag is being executed.

SW_LOCATION
Defines the location of the product, which may have been changed from the default product directory. When combined with the **SW_ROOT_DIRECTORY**, this variable tells scripts where the product files are located.

SW_PATH
A PATH variable which defines a minimum set of commands available for use in a control script (e.g. **/sbin:/usr/bin**).

SW_ROOT_DIRECTORY
Defines the root directory in which the session is operating, either "/" or an alternate root directory. This variable tells control scripts the root directory in which the products are installed. A script must use this directory as a prefix to **SW_LOCATION** to locate the product's installed files. The configure script is only run when **SW_ROOT_DIRECTORY**is **/**.

SW_SESSION_OPTIONS
Contains the pathname of a file containing the value of every option for a particular command, including software and target selections. This lets scripts retrieve any command options and values other than the ones provided explicitly by other environment variables. For example, when the file pointed to by **SW_SESSIONS_OPTIONS** is made available to a request script, the *targets* option contains a list of *software_collection_specs* for all targets specified for the command. When the file pointed to by **SW_SESSIONS_OPTIONS** is made available to other scripts, the *targets* option contains the single

software_collection_spec for the targets on which the script is being executed.

SW_SOFTWARE_SPEC
This variable contains the fully qualified software specification of the current product or fileset. The software specification allows the product or fileset to be uniquely identified.

RETURN VALUES

swask returns one of these codes:

- **0** Command successful on all targets
- **1** Command failed on all targets
- **2** Command failed on some targets

DIAGNOSTICS

The **swask** command writes to stdout, stderr, and to the **swask** logfile.

Standard Output

An interactive **swask** session does not write to stdout. A non-interactive **swask** session writes messages for significant events. These include:

- a begin and end session message,
- selection, analysis, and execution task messages for each *target_selection*.

Standard Error

An interactive **swask** session does not write to stderr. A non-interactive **swask** session writes messages for all WARNING and ERROR conditions to stderr.

Logging

Both interactive and non-interactive **swask** sessions log summary events at the host where the command was invoked. They log detailed events to the **swask.log** logfile associated with each *target_selection*.

Command Log

The **swask** command logs all stdout and stderr messages to the the logfile **/var/adm/sw/swask.log**. Similar messages are logged by an interactive **swask** session. You can specify a different logfile by modifying the **logfile** option.

EXAMPLES

Run all request scripts from the default depot (**/var/spool/sw**) depot and write the response file (named **response**) back to the same depot:

```
swask -s /var/spool/sw \*
```

Run the request script for **Product1** from depot **/tmp/sample.depot.1** on remote host **swposix**, create the catalog **/tmp/test1.depot** on the local controller machine, and place the response file (named **response**) in the catalog:

```
swask -s swposix:/tmp/sample.depot.1 -c /tmp/test1.depot Product1
```

Run request scripts from remote depot **/tmp/sample.depot.1** on host **swposix** only when a response file is absent, create the catalog **/tmp/test1.depot** on the local controller machine, and place the response file (named **response**) in the catalog:

```
swask -s swposix:/tmp/sample.depot.1 -c /tmp/test1.depot
-x ask=as_needed \*
```

FILES

$HOME/.swdefaults
Contains the user-specific default values for some or all SD options. If this file does not exist, SD looks for user-specific defaults in **$HOME/.sw/defaults**.

$HOME/.sw/sessions/
Contains session files automatically saved by the SD commands or explicitly saved by the user.

/usr/lib/sw/sys.defaults
Contains the master list of current SD options, with their default values, for documentation purposes only.

/var/adm/sw/
The directory which contains all of the configurable (and non-configurable) data for SD. This

directory is also the default location of log files.

/var/adm/sw/defaults
Contains the active system-wide default values for some or all SD options.

/var/adm/sw/products/
The Installed Products Database (IPD), a catalog of all products installed on a system.

/var/adm/sw/swask.log
Contains all stdout and stderr messages generated by **swask**.

AUTHOR

swask was developed by the Hewlett-Packard Company.

SEE ALSO

swconfig(1M), swinstall(1M), sd(5).

Software Distributor Administration Guide, available at **http://docs.hp.com**.

SD customer web site at **http://software.hp.com/SD_AT_HP/**.

NAME

swconfig - configure, unconfigure, or reconfigure installed software

SYNOPSIS

swconfig [**-p**] [**-u**] [**-v**] [**-c** *catalog*] [**-C** *session_file*] [**-f** *software_file*] [**-J** *jobid*] [**-Q** *date*] [**-S** *session_file*] [**-t** *target_file*] [**-x** *option=value*] [**-X** *option_file*] [*software_selections*] [**@** *target_selections*]

Remarks

- This command supports operation on remote systems. See **Remote Operation** below.
- **swconfig** can perform limited interactive operations. See **Interactive Operation** below.
- For an overview of all SD commands, see the *sd*(5) man page by typing **man 5 sd** on the command line.

DESCRIPTION

The **swconfig** command configures, unconfigures, or reconfigures installed software products for execution on the specified targets. The **swconfig** command transitions software between INSTALLED and CONFIGURED states. Although software is automatically configured as part of the **swinstall** command and unconfigured as part of the **swremove** command, **swconfig** lets you configure or unconfigure software independently when the need arises.

Configuration primarily involves the execution of vendor-supplied *configure scripts*. These scripts perform configuration tasks which enable the use of the software on the target hosts. A vendor can also supply *unconfigure scripts* to "undo" the configuration performed by the configure script.

NOTES:

- You should execute **swconfig** when an initial configuration by **swinstall** failed, was deferred, or needs to be changed.
- With **swinstall**, you can defer configuration by using the **defer_configure** default option.
- **swinstall** does not perform configuration on multiple versions of software.
- The **swconfig** command only operates on software installed to the primary root file system.
- **swinstall** and **swremove** do not run configure or unconfigure scripts when you specify an alternate root directory with those commands.

Other features of **swconfig** include:

- By default, the **swconfig** command supports only configuration of compatible software.
- If a fileset specifies a prerequisite on other software, that software must be in a "configured" state before the software specifying the dependency will be configured.
- The **swconfig** command configures multiple versions of a product if you set **allow_multiple_versions=true**. The vendor must therefore detect and prevent multiple configured versions in their configure scripts, if that is necessary.
- Configure scripts are useful for software updates and reinstallation, as well as first-time installation.

Remote Operation

You can enable Software Distributor (SD) to manage software on remote systems. To let the root user from a central SD *controller* (also called the *central management server* or *manager node*) perform operations on a remote *target* (also called the *host* or *agent*):

1) Install a special HP ServiceControl Manager fileset on the remote systems. This enables remote operations by automatically setting up the root, host, and template Access Control Lists (ACLs) on the remote machines and permitting root access from the controller system. To install the fileset, run the following command on each remote system:

swinstall -s *controller***:/var/opt/mx/depot11 AgentConfig.SD-CONFIG**

NOTES:

- *controller* is the name of the central management server.

- If the target is running HP-UX 10.20, use the same command but substitute **depot10** for **depot11**.
- Targets previously set up by SD/OV to be managed by this controller do not need this step.
- SD does not require any other ServiceControl Manager filesets.

2) (Optional) Enable the GUI interfaces for remote operations by creating the **.sdkey** file on the controller. Use this command:

```
touch /var/adm/sw/.sdkey
```

(This step is not required when you use SD from within the HP ServiceControl Manager.)

NOTE: You can also set up remote access by using the *swacl*(1M) command directly on the remote machines to grant root or non-root access to users from the controller system.

Interactive Operation

swconfig can perform limited interactive operations when the **ask** option is set to **true**. This option executes an interactive *request script*. Request scripts can also be executed by **swinstall** and **swask**. See the **ask=false** default option for more information. See also *swinstall*(1M) and *swask*(1M).

Options

swconfig supports the following options:

-c *catalog* Specifies the pathname of an exported catalog which stores copies of the response file or files created by a request script (if **-x ask=true** or **-x ask=as_needed**). Response files are also stored in the **Installed Products Database**.

-C *session_file*
Save the current options and operands to *session_file*. You can enter a relative or absolute path with the file name. The default directory for session files is **$HOME/.sw/sessions/**. You can recall a session file with the **-S** option.

-f *software_file*
Read the list of *software_selections* from *software_file* instead of (or in addition to) the command line.

-J *jobid* Executes the previously scheduled job. This is the syntax used by the daemon to start the job.

-p Previews a configuration task by running the session through the analysis phase only.

-Q *date* Schedules the job for this date. You can change the date format by editing the **/var/adm/sw/getdate.templ** file.

-S *session_file*
Execute **swconfig** based on the options and operands saved from a previous session, as defined in *session_file*. You can save session information to a file with the **-C** option.

-t *target_file* Read the list of *target_selections* from *target_file* instead of (or in addition to) the command line.

-u Causes **swconfig** to unconfigure the software instead of configuring it.

-v Turns on verbose output to stdout. (The **swconfig** logfile is not affected by this option.) Verbose output is enabled by default; see the **verbose** option below.

-x *option=value*
Set the session *option* to *value* and override the default value (or a value in an alternate *option_file* specified with the **-X** option). Multiple **-x** options can be specified.

-X *option_file* Read the session options and behaviors from *option_file*.

Operands

Most SD commands support two types of operands: *software selections* followed by *target selections*. These operands are separated by the "@" (at) character. This syntax implies that the command operates on "software selections at targets".

Software Selections

The **`swconfig`** command supports the following syntax for each *software_selection*:

bundle[**.***product*[**.***subproduct*][**.***fileset*]][**,***version*]

product[**.***subproduct*][**.***fileset*][**,***version*]

- The **`=`** (equals) relational operator lets you specify selections with the following shell wildcard and pattern-matching notations:

  `[ ]`, **`*`**, **`?`**

- *Bundles* and *subproducts* are recursive. *Bundles* can contain other *bundles* and *subproducts* can contain other *subproducts*.
- The **`\*`** software specification selects all products. Use this specification with caution.

The **`version`** component has the form:

[**`,r`** *<op> revision*][**`,a`** *<op> arch*][**`,v`** *<op> vendor*]
[**`,c`** *<op> category*][**`,q=`***qualifier*][**`,l=`***location*]
[**`,fr`** *<op> revision*][**`,fa`** *<op> arch*]

- *location* applies only to installed software and refers to software installed to a location other than the default product directory.
- **`fr`** and **`fa`** apply only to filesets.
- The *<op>* (relational operator) component can be of the form:

  `=`, **`==`**, **`>=`**, **`<=`**, **`<`**, **`>`**, or **`!=`**

  which performs individual comparisons on dot-separated fields.

  For example, **`r>=B.10.00`** chooses all revisions greater than or equal to **`B.10.00`**. The system compares each dot-separated field to find matches.

- The **`=`** (equals) relational operator lets you specify selections with the shell wildcard and pattern-matching notations:

  `[ ]`, **`*`**, **`?`**, **`!`**

  For example, the expression **`r=1[01].*`** returns any revision in version 10 or version 11.

- All version components are repeatable within a single specification (e.g. **`r>=A.12`**, **`r<A.20`**). If multiple components are used, the selection must match all components.
- Fully qualified software specs include the **`r=`**, **`a=`**, and **`v=`** version components even if they contain empty strings.
- No space or tab characters are allowed in a software selection.
- The software *instance_id* can take the place of the version component. It has the form:

  [*instance_id*]

  within the context of an exported catalog, where *instance_id* is an integer that distinguishes versions of products and bundles with the same tag.

S

The **`\*`** software specification selects all products. It is not allowed when removing software from the root directory **`/`**.

Target Selections

`swconfig`
supports this syntax for each *target_selection*.

[*host*][**`:`**][**`/`***directory*]

The **`:`** (colon) is required if both a host and directory are specified.

EXTERNAL INFLUENCES

Default Options

In addition to the standard options, several SD behaviors and policy options can be changed by editing the default values found in:

/var/adm/sw/defaults the system-wide default values.

$HOME/.swdefaults the user-specific default values.

Values must be specified in the defaults file using this syntax:

[*command_name*.]*option*=*value*

The optional *command_name* prefix denotes one of the SD commands. Using the prefix limits the change in the default value to that command. If you leave the prefix off, the change applies to all commands.

You can also override default values from the command line with the **-x** or **-X** options:

command **-x** *option*=*value*

command **-X** *option_file*

The following section lists all of the keywords supported by the **swlist** commands. If a default value exists, it is listed after the "=".

The policy options that apply to **swconfig** are:

admin_directory=/var/adm/sw (for normal mode)
admin_directory=/var/home/LOGNAME/sw (for nonprivileged mode)

The location for SD logfiles and the default parent directory for the installed software catalog. The default value is **/var/adm/sw** for normal SD operations. When SD operates in nonprivileged mode (that is, when the **run_as_superuser** default option is set to **true**):

- The default value is forced to **/var/home/LOGNAME/sw**.
- The path element **LOGNAME** is replaced with the name of the invoking user, which SD reads from the system password file.
- If you set the value of this option to **HOME/***path*, SD replaces **HOME** with the invoking user's home directory (from the system password file) and resolves *path* relative to that directory. For example, **HOME/my_admin** resolves to the **my_admin** directory in your home directory.
- If you set the value of the **installed_software_catalog** default option to a relative path, that path is resolved relative to the value of this option.

SD's nonprivileged mode is intended only for managing applications that are specially designed and packaged. This mode cannot be used to manage the HP-UX operating system or patches to it. For a full explanation of nonprivileged SD, see the *Software Distributor Administration Guide*, available at the **http://docs.hp.com** web site.

See also the **installed_software_catalog** and **run_as_superuser** options.

agent_auto_exit=true

Causes the target agent to automatically exit after Execute phase, or after a failed Analysis phase. This is forced to **false** when the controller is using an interactive UI, or when -p (preview) is used. This enhances network reliability and performance. The default is **true** - the target agent will automatically exit when appropriate. If set to **false**, the target agent will not exit until the controller ends the session.

agent_timeout_minutes=10000

Causes a target agent to exit if it has been inactive for the specified time. This can be used to make target agents more quickly detect lost network connections since RPC can take as long as 130 minutes to detect a lost connection. The recommended value is the longest period of inactivity expected in your environment. For command line invocation, a value between 10 minutes and 60 minutes is suitable. A value of 60 minutes or more is recommended when the GUI will be used. The default of 10000 is slightly less than 7 days.

allow_incompatible=false

Requires that the software products which are being configured be "compatible" with the target selections. (All of the target selections must match the list of supported systems defined for each selected product.) If set to **true**, target compatibility is not enforced.

allow_multiple_versions=false

Prevents the configuration of another, independent version of a product when a version already is configured at the target.

If set to **true**, another version of an existing product can be configured in its new location. Multiple versions can only be installed if a product is locatable. Multiple configured versions will not work unless the product supports it.

ask=false

When **ask=true**, executes a **request script**, which asks for a user response. If **ask=as_needed**, the **swask** command first determines if a response file already exists in the control directory and executes the **request** script only when a response file is absent.

If set to **ask=true**, or **ask=as_needed**, you can use the **-c** *catalog* option to specify the pathname of an exported catalog to store copies of the response file or files created by the **request** script.

See *swask*(1M) for more information on **request** scripts.

autoremove_job=false

Controls automatic job removal of completed jobs. If the job is automatically removed, job information (job status or controller/agent logfiles) cannot be queried with **swjob**.

autoselect_dependencies=true

Controls the automatic selection of prerequisite, corequisite, and exrequisite software that is not explicitly selected by the user. This option does not apply to **swconfig -u**. The default is: **true**. The requisite software will be automatically selected for configuration. Specifying **false** causes requisite software, which is not explicitly selected, to not be automatically selected for configuration.

autoselect_dependents=false

Controls the automatic selection of dependent software that is not explicitly selected by the user. A dependent is the opposite of a requisite. A dependent fileset has established either a prerequisite or a corequisite on the selected fileset. Specifying **true** causes dependent software to be automatically selected for unconfiguration. The default, **false** causes dependent software, which is not explicitly selected, to not be automatically selected for unconfiguration.

compress_index=false

Determines whether SD commands create compressed INDEX and INFO catalog files when writing to target depots or roots. The default of **false** does not create compressed files. When set to **true**, SD creates compressed and uncompressed INDEX and INFO files. The compressed files are named INDEX.gz and INFO.gz, and reside in the same directories as the uncompressed files.

Compressed files can enhance performance on slower networks, although they may increase disk space usage due to a larger Installed Products Database and depot catalog. SD controllers and target agents for HP-UX 11.01 and higher automatically load the compressed INDEX and INFO files from the source agent when:

- The source agent supports this feature.
- INDEX.gz or INFO.gz exist on the source depot.
- INDEX.gz or INFO.gz are not older than the corresponding uncompressed INDEX or INFO files.

The uncompressed INDEX or INFO file is accessed by the source agent if any problem occurs when accessing, transferring, or uncompressing the INDEX.gz or INFO.gz file.

controller_source

Location of a depot for the controller to access to resolve selections. This has no effect on which sources the target uses. Specify this as host, /path, or host:/path. Useful for reducing network traffic between controller and target.

enforce_dependencies=true

Requires that all dependencies specified by the *software_selections* be resolved at the *target_selections*.

The **swconfig**, command will not proceed unless the dependencies have also been selected or already exist at the target in the correct state (INSTALLED or CONFIGURED). This prevents unusable software from being configured on the system.

If set to **false**, dependencies will still be checked, but not enforced. Corequisite dependencies, if not enforced, may keep the selected software from working properly. Prerequisite and exrequisite dependencies, if not enforced, may cause the configuration to fail.

enforce_scripts=true
Controls the handling of errors generated by scripts. If **true**, and the vendor-supplied script returns an error, the configure or unconfigure operation stops. An error message appears reporting that the execution phase failed. If **false**, **swconfig** attempts to continue operation. A warning message appears reporting that the execution succeeded.

installed_software_catalog=products
Defines the directory path where the Installed Products Database (IPD) is stored. This information describes installed software. When set to an absolute path, this option defines the location of the IPD. When this option contains a relative path, the SD controller appends the value to the value specified by the **admin_directory** option to determine the path to the IPD. For alternate roots, this path is resolved relative to the location of the alternate root. This option does not affect where software is installed, only the IPD location.

This option permits the simultaneous installation and removal of multiple software applications by multiple users or multiple processes, with each application or group of applications using a different IPD.

Caution: use a specific **installed_software_catalog** to manage a specific application. SD does not support multiple descriptions of the same application in multiple IPDs.

See also the **admin_directory** and **run_as_superuser** options, which control SD's nonprivileged mode. (This mode is intended only for managing applications that are specially designed and packaged. This mode cannot be used to manage the HP-UX operating system or patches to it. For a full explanation of nonprivileged SD, see the *Software Distributor Administration Guide*, available at the **http://docs.hp.com** web site.)

job_title=
This is an ASCII string giving a title to a job. It is displayed along with the job ID to provide additional identifying information about a job when **swjob** is invoked.

log_msgid=0
Adds numeric identification numbers at the beginning of SD logfile messages:
0 (default) No identifiers are attached to messages.
1 Adds identifiers to ERROR messages only.
2 Adds identifiers to ERROR and WARNING messages.
3 Adds identifiers to ERROR, WARNING, and NOTE messages.
4 Adds identifiers to ERROR, WARNING, NOTE, and certain other informational messages.

logdetail=false
Controls the amount of detail written to the logfile. When set to **true**, this option adds detailed task information (such as options specified, progress statements, and additional summary information) to the logfile. This information is in addition to log information controlled by the **loglevel** option.

See **loglevel** below and the *sd*(5) manual page, by typing **man**5**sd**, for more information.

logfile=/var/adm/sw/swconfig.log
This is the default command log file for the **swconfig** command.

loglevel=1
Controls the log level for the events logged to the command logfile, the target agent logfile, and the source agent logfile. This information is in addition to the detail controlled by the **logdetail** option. (See **logdetail** above and the *sd*(5) manual page, by typing **man 5 sd** , for more information.) A value of

0 provides no information to the logfile.
1 enables verbose logging to the logfiles.
2 enables very verbose logging to the logfiles.

mount_all_filesystems=true
By default, the **swconfig** command attempts to automatically mount all filesystems in

S

the **/etc/fstab** file at the beginning of the analysis phase, to ensure that all listed filesystems are mounted before proceeding. This policy helps to ensure that files are not loaded into a directory that may be below a future mount point.

If set to **false**, the mount operation is not attempted, and no check of the current mounts is performed.

reconfigure=false
Prevents software which is already in the CONFIGURED state from being reconfigured. If set to **true**, CONFIGURED software can be reconfigured.

rpc_binding_info=ncacn_ip_tcp:[2121] ncadg_ip_udp:[2121]
Defines the protocol sequence(s) and endpoint(s) on which the daemon listens and on which the other commands use to contact the daemon. If the connection fails for one protocol sequence, the next is attempted. SD supports both the tcp (ncacn_ip_tcp:[2121]) and udp (ncadg_ip_udp:[2121]) protocol sequence on most platforms.

rpc_timeout=5
Relative length of the communications timeout. This is a value in the range from 0 to 9 and is interpreted by the DCE RPC. Higher values mean longer times; you may need a higher value for a slow or busy network. Lower values will give faster recognition on attempts to contact hosts that are not up, or are not running the **swagentd**. Each value is approximately twice as long as the preceding value. A value of 5 is about 30 seconds for the **ncadg_ip_udp** protocol sequence.

run_as_superuser=true
This option controls SD's nonprivileged mode. This option is ignored (treated as true) when the invoking user is super-user.

When set to the default value of true, SD operations are performed normally, with permissions for operations either granted to a local super-user or set by SD ACLs. (See *swacl*(1M) for details on ACLs.)

When set to false and the invoking user is local and is *not* super-user, nonprivileged mode is invoked:

- Permissions for operations are based on the user's file system permissions.
- SD ACLs are ignored.
- Files created by SD have the uid and gid of the invoking user, and the mode of created files is set according to the invoking user's umask.

SD's nonprivileged mode is intended only for managing applications that are specially designed and packaged. This mode cannot be used to manage the HP-UX operating system or patches to it. For a full explanation of nonprivileged SD, see the *Software Distributor Administration Guide*, available at the **http://docs.hp.com** web site.

See also the **admin_directory** and **installed_software_catalog** options.

select_local=true
If no *target_selections* are specified, select the local host as the target of the command.

software=
Defines the default *software_selections*. There is no supplied default. If there is more than one software selection, they must be separated by spaces.

targets=
Defines the default *target_selections*. There is no supplied default (see **select_local** above). If there is more than one target selection, they must be separated by spaces.

verbose=1
Controls the verbosity of the output (stdout). A value of
0 disables output to stdout. (Error and warning messages are always written to stderr).
1 enables verbose messaging to stdout.

write_remote_files=false
Prevents the configuring of files on a target which exists on a remote (NFS) filesystem. All files on a remote filesystem will be skipped.

If set to **true** and if the superuser has write permission on the remote filesystem, the remote files will not be skipped, but will be configured.

Session File

Each invocation of the **swconfig** command defines a configuration session. The invocation options, source information, software selections, and target hosts are saved before the installation or copy task actually commences. This lets you re-execute the command even if the session ends before proper completion.

Each session is automatically saved to the file **$HOME/.sw/sessions/swremove.last**. This file is overwritten by each invocation of **swconfig**.

You can also save session information to a specific file by executing **swconfig** with the **-C** *session__file* option.

A session file uses the same syntax as the defaults files. If you do not specify a specific path for the session file, the default location is **$HOME/.sw/sessions/**.

To re-execute a session file, specify the session file as the argument for the **-S** *session__file* option of **swconfig**.

Note that when you re-execute a session file, the values in the session file take precedence over values in the system defaults file. Likewise, any command line options or parameters that you specify when you invoke **swconfig** take precedence over the values in the session file.

Environment Variables

The environment variable that affects the **swconfig** command is:

LANG Determines the language in which messages are displayed. If LANG is not specified or is set to the empty string, a default value of **C** is used. See *lang*(5) for more information.

NOTE: The language in which the SD agent and daemon log messages are displayed is set by the system configuration variable script, **/etc/rc.config.d/LANG**. For example, **/etc/rc.config.d/LANG**, must be set to **LANG=ja_JP.SJIS** or **LANG=ja_JP.eucJP** to make the agent and daemon log messages display in Japanese.

LC_ALL Determines the locale to be used to override any values for locale categories specified by the settings of **LANG** or any environment variables beginning with **LC_**.

LC_CTYPE
Determines the interpretation of sequences of bytes of text data as characters (e.g., single- versus multibyte characters in values for vendor-defined attributes).

LC_MESSAGES
Determines the language in which messages should be written.

LC_TIME
Determines the format of dates (*create_date* and *mod_date*) when displayed by **swlist**. Used by all utilities when displaying dates and times in **stdout**, **stderr**, and **logging**.

TZ Determines the time zone for use when displaying dates and times.

Environment variables that affect scripts are:

SW_CATALOG
Holds the path to the Installed Products Database (IPD), relative to the path in the **SW_ROOT_DIRECTORY** environment variable. Note that you can specify a path for the IPD using the **installed_software_catalog** default option.

SW_CONTROL_DIRECTORY
Defines the current directory of the script being executed, either a temporary catalog directory, or a directory within in the Installed Products Database (IPD). This variable tells scripts where other control scripts for the software are located (e.g. subscripts).

SW_CONTROL_TAG
Holds the tag name of the control_file being executed. When packaging software, you can define a physical name and path for a control file in a depot. This lets you define the control_file with a name other than its tag and lets you use multiple control file definitions to point to the same file. A control_file can query the **SW_CONTROL_TAG** variable to determine which tag is being executed.

SW_LOCATION
Defines the location of the product, which may have been changed from the default product directory. When combined with the **SW_ROOT_DIRECTORY**, this variable tells scripts where the product files are located.

SW_PATH
A PATH variable which defines a minimum set of commands available for use in a control script (e.g. **/sbin:/usr/bin**).

SW_ROOT_DIRECTORY
Defines the root directory in which the session is operating, either "/" or an alternate root directory. This variable tells control scripts the root directory in which the products are installed. A script must use this directory as a prefix to **SW_LOCATION** to locate the product's installed files. The configure script is only run when **SW_ROOT_DIRECTORY**is **/**.

SW_SESSION_OPTIONS
Contains the pathname of a file containing the value of every option for a particular command, including software and target selections. This lets scripts retrieve any command options and values other than the ones provided explicitly by other environment variables. For example, when the file pointed to by **SW_SESSIONS_OPTIONS** is made available to a **request** script, the *targets* option contains a list of *software_collection_specs* for all targets specified for the command. When the file pointed to by **SW_SESSIONS_OPTIONS** is made available to other scripts, the *targets* option contains the single *software_collection_spec* for the targets on which the script is being executed.

SW_SOFTWARE_SPEC
This variable contains the fully qualified software specification of the current product or fileset. The software specification allows the product or fileset to be uniquely identified.

Signals

The **swconfig** command catches the signals SIGQUIT and SIGINT, and SIGUSR1. If these signals are received, **swconfig** prints a message, sends a Remote Procedure Call (RPC) to the agents to wrap up, and then exits.

The agent ignores SIGHUP, SIGINT, and SIGQUIT. It immediately exits gracefully after receiving SIGTERM, SIGUSR1, or SIGUSR2. Killing the agent may leave corrupt software on the system, and thus should only be done if absolutely necessary. Note that when an SD command is killed, the agent does not terminate until completing the task in progress.

The daemon ignores SIGHUP, SIGINT and SIGQUIT. It immediately exits gracefully after receiving SIGTERM and SIGUSR2. After receiving SIGUSR1, it waits for completion of a copy or remove from a depot session before exiting, so that it can register or unregister depots if necessary. Requests to start new sessions are refused during this wait.

Each agent will complete the configuration task (if the execution phase has already started) before it wraps up. This avoids leaving software in a corrupt state.

RETURN VALUES

The **swconfig** command returns:

0 The *software_selections* were successfully configured.
1 The configure operation failed on all *target_selections*.
2 The configure operation failed on some *target_selections*.

DIAGNOSTICS

The **swconfig** command writes to stdout, stderr, and to specific logfiles.

Standard Output

The **swconfig** command writes messages for significant events. These include:

- a begin and end session message,
- selection, analysis, and execution task messages for each *target_selection*.

Standard Error

The **swconfig** command also writes messages for all WARNING and ERROR conditions to stderr.

Logging

The **swconfig** command logs summary events at the host where the command was invoked. It logs detailed events to the **swagent** logfile associated with each *target_selection*.

Command Log

The **swconfig** command logs all stdout and stderr messages to the the logfile **/var/adm/sw/swconfig.log**. (The user can specify a different logfile by modifying the **log-file** option.)

Target Log

A **swagent** process performs the actual configure operation at each *target_selection*. The **swagent** logs events to the file **/var/adm/sw/swagent.log**. You can view the command and target log files with the **swjob**or**sd** command.

EXAMPLES

Configure the C and Pascal products on the local host:

```
swconfig cc pascal
```

Configure *Product1*, use any associated response files generated by a request script, and save response files under **/tmp/resp1**:

```
swconfig -x ask=true -c /tmp/resp1 Product1
```

Reconfigure the HP Omniback product:

```
swconfig -x reconfigure=true Omniback
```

Configure the version of HP Omniback that was installed at **/opt/Omniback_v2.0**:

```
swconfig Omniback,l=/opt/Omniback_v2.0
```

Unconfigure the *software_selections* listed in the file **/tmp/install.products** on the hosts listed in the file **/tmp/install.hosts**:

```
swconfig -u -f /tmp/install.products -t /tmp/install.hosts
```

Configure the C and Pascal products on remote hosts:

```
swconfig cc pascal @  hostA hostB hostC
```

LIMITATIONS

The SD-UX version of **swconfig** does not support the configuration, unconfiguration, or reconfiguration of installed software on remote targets.

FILES

$HOME/.swdefaults
: Contains the user-specific default values for some or all SD software management command options.

$HOME/.sw/sessions/
: Contains session files automatically saved by the SD software management commands, or explicitly saved by the user.

/usr/lib/sw/sys.defaults
: Contains the master list of current SD options with their default values.

/var/adm/sw/
: The directory which contains all configurable and non-configurable data for SD software management commands. This directory is also the default location of logfiles.

/var/adm/sw/defaults
: Contains the active system-wide default values for some or all SD software management command options.

/var/adm/sw/getdate.templ
: Contains the set of date/time templates used when scheduling jobs.

/var/adm/sw/products/
: The Installed Products Database (IPD), a catalog of all products installed on a system.

AUTHOR

swconfig was developed by the Hewlett-Packard Company.

SEE ALSO

swpackage(4), swacl(1M), swagentd(1M), swask(1M), swcopy(1M), swinstall(1M), swjob(1M), swlist(1M), swmodify(1M), swpackage(1M), swreg(1M), swremove(1M), swverify(1M) install-sd(1M), sd(4), sd(5).

Software Distributor Administration Guide, available at **http://docs.hp.com**.

SD customer web site at **http://software.hp.com/SD_AT_HP/**.

NAME

swinstall, swcopy - install and configure software products; software products for subsequent installation or distribution; respectively

SYNOPSIS

swinstall [*XToolkit Options*] [**-i**] [**-p**] [**-r**] [**-v**] [**-c** *catalog*] [**-C** *session_file*] [**-f** *software_file*] [**-J** *jobid*] [**-Q** *date*] [**-s** *source*] [**-S** *session_file*] [**-t** *target_file*] [**-x** *option=value*] [**-X** *option_file*] [*software_selections*] [**@** *target_selections*]

swcopy [*XToolkit Options*] [**-i**] [**-p**] [**-v**] [**-C** *session_file*] [**-f** *software_file*] [**-J** *jobid*] [**-Q** *date*] [**-s** *source*] [**-S** *session_file*] [**-t** *target_file*] [**-x** *option=value*] [**-X** *option_file*] [*software_selections*] [**@** *target_selections*]

Remarks

- This command supports operation on remote systems. See the **Remote Operation** section below for details.
- **swinstall** and **swcopy** support an interactive user interface that can be invoked alone or by the **sd** command. See **Interactive Operation** below.
- For an overview of all SD commands, see the *sd*(5) man page by typing **man 5 sd** on the command line.

DESCRIPTION

The **swinstall** command installs the *software_selections* from a software *source* to either the local host or to one or more *target_selections* (root filesystems). By default, the software is configured for use on the target after it is installed. (The software is not configured when installed into an alternate root directory.)

The **swcopy** command copies or merges *software_selections* from a software *source* to one or more software depot *target_selections*. These depots can then be accessed as a software source by the **swinstall** command.

Remote Operation

You can enable Software Distributor (SD) to manage software on remote systems. To let the root user from a central SD *controller* (also called the *central management server* or *manager node*) perform operations on a remote *target* (also called the *host* or *agent):*

1) Install a special HP ServiceControl Manager fileset on the remote systems. This enables remote operations by automatically setting up the root, host, and template Access Control Lists (ACLs) on the remote machines and permitting root access from the controller system. To install the fileset, run the following command on each remote system:

swinstall -s *controller* **:/var/opt/mx/depot11 AgentConfig.SD-CONFIG**

NOTES:

- *controller* is the name of the central management server.
- If the target is running HP-UX 10.20, use the same command but substitute **depot10** for **depot11**.
- Targets previously set up by SD/OV to be managed by this controller do not need this step.
- SD does not require any other ServiceControl Manager filesets.

2) (Optional) Enable the GUI interfaces for remote operations by creating the **.sdkey** file on the controller. Use this command:

touch /var/adm/sw/.sdkey

(This step is not required when you use SD from within the HP ServiceControl Manager.)

NOTE: You can also set up remote access by using the *swacl*(1M) command directly on the remote machines to grant root or non-root access to users from the controller system.

Interactive Operation

swinstall and **swcopy** each support a graphical user interface (GUI). (If your terminal or display cannot support the GUI, these commands also provide a terminal user interface, in which screen navigation is done with the keyboard and no mouse.)

S

To invoke the GUI, enter

```
swinstall
```

or

```
swcopy
```

on the command line (without any command-line options).

You can also invoke the GUI by including the **-i** option with any other command-line options.

The **sd** command provides an interactive interface for monitoring and scheduling software jobs. You can also use **sd** to invoke the **swinstall**, **copy**, and **swremove** GUIs.

If you have enabled SD's remote operations features, **swinstall**, **swcopy**, and **swremove** provide enhanced GUIs to support operations on remote targets. See **Remote Operation** above for details about enabling remote operations and the enhanced GUIs.

The command-line version of **swinstall** can also function interactively when the **ask** option is set to **true**. This option executes an interactive *request script*. Request scripts can also be executed by **swconfig** and **swask**. See *swconfig*(1M)*and swask*(1M), and the **ask=false** default option for more information.

Updating the Operating System

To perform an OS update, HP recommends that you first use the **update-ux** command to perform automated update preparation checks and to get the newest version of **swinstall** from media. See *update-ux*(1M) for more information.

Reinstalling SD

If your copy of SD is unusable, or if you want to install a newer version of SD, HP recommends that you use the **install-sd** command. This command reinstalls SD and also installs any SD patches that exist in the source depot. See *install-sd*(1M) for more information.

Installing Kernel Software

In HP-UX, the kernel installation process requires that the system boots using the kernel at **/stand/vmunix**. Make sure that your system is booted to the **/stand/vmunix** kernel before you install any kernel software or perform an operating system update.

Dependencies Between Software

The **swinstall** command supports *dependencies*, which is software that must be present or absent before or during the installation of another piece of software. Dependencies apply between filesets and other filesets and products. SD supports three types of dependencies: *prerequisites* that must be installed and configured before the dependent fileset is installed and configured (respectively); *corequisites* that must be installed and configured before the dependent is usable. *exrequisites* that prevent a dependent fileset from being installed or configured when they are present.

S

If a *software_selection* specifies a dependency on other filesets and/or products, **swinstall** automatically select that software.

By default, all dependencies must be resolved before **swinstall** can proceed. You can override this policy using the **enforce_dependencies** option.

Note that if you specify a dependency for a fileset and the fileset is superseded by another fileset as part of a patch, **swinstall** still recognizes the dependency.

Features and Differences between swinstall and swcopy

The key difference between **swinstall** and **swcopy** is that **swinstall** performs the software installation, while **swcopy** copies software into a depot, making it available as a source for installation by **swinstall**.

NOTE: To copy to a tape, see the *swpackage*(1M) manpage.

Other features (differences) include:

- The **swinstall** command executes several vendor-supplied scripts during the installation and configuration of the *software_selections*. The **swcopy** command does not execute these scripts. The **swinstall** command supports the following scripts:

`request` a script that asks the user questions and stores responses in a **response** file. The response file can then be used by configuration or other scripts.

`checkinstall`
a script executed during the analysis of a **`target_selection`**, it checks that the installation can be attempted. If this check fails, the software product is not installed.

`preinstall` a script executed immediately before the software's files are installed.

`postinstall`
a script executed immediately after the software's files are installed.

`configure` a script executed during the configuration of a *target_selection*, it configures the target for the software (and the software for the target). The **`preinstall`** and **`postinstall`** scripts are not intended to be used for configuration tasks. They are to be used for simple file management needs such as removing obsolete files from the previous revision (which was just updated).

`unpreinstall`
a script executed immediately after the software's actual files are restored if the software install will fail and the **`autorecover_product`** option is set to **`true`**. The script undoes the steps performed by **`preinstall`** script.

`unpostinstall`
a script executed immediately before the software's actual files are restored if the software install failed and the **`autorecover_product`** option is set to **`true`**. The script undoes the steps performed by **`postinstall`** script.

- When a depot is created or modified using **`swcopy`**, **catalog files** are built that describe the depot (comparable to the **Installed Products Database** (IPD) files that are built by the **`swinstall`** command).
- By default, the **`swinstall`** command only allows the selection of compatible software from the *source*. This constraint ensures that the architecture of the software matches that of the *target_selections*. No compatibility checks are performed by the **`swcopy`** command. (A depot can be a repository of software targeted for a variety of architectures and operating systems.)
- By default, **`swinstall`** supports updates to higher revisions of software. If a *software_selection* of the same revision is already installed, **`swinstall`** will not reinstall it. If a *software_selection* has a lower revision than the same software which is already installed, **`swinstall`** will not reinstall it. (The user can override these behaviors with control options.)
- The **`swinstall`** command creates hard links and symbolic links as specified for the software. If it encounters a symbolic link where it expected a regular file, **`swinstall`** follows the symbolic link and updates the file to which it points.
- The **`swinstall`** command does not remove a product's current files before installing the new ones. A fileset's install scripts can do that, if necessary. Files being replaced are overwritten unless they are in use. If in use, they are unlinked or moved to #<file>. If the *autorecover_product* option is set to **`true`**; all files are saved to #<file>, and restored if the install fails.
- The **`swinstall`** command supports kernel building scripts and rebooting. Before or after software that modifies the kernel is installed or updated, **`swinstall`** executes system-specific scripts to prepare for or build the new version of the kernel. The remaining *software_selections* are then installed. These scripts are defined in **`swagent`** options and include: **`install_setup_cmd`**, **`system_prep_cmd`**, **`kernel_build_cmd`**, and **`install_cleanup_cmd`**.

  After software that requires a system reboot is installed or updated, **`swinstall`** automatically reboots the system. The reboot command is defined by the **`swagent`** option: **`reboot_cmd`**.

  When updating the operating system (see *update-ux*(1M) for more information.), you should install kernel software first to ensure that a new kernel can be generated before the rest of the operating system is updated. After all the *software_selections* are updated or installed, **`swinstall`** reboots using the new kernel, then executes the configure scripts for each *software_selection*. After these scripts complete, it reboots the system again to restore it to its normal state.
- No kernel building or system reboots are performed by **`swcopy`**.

- Both the **swinstall** and **swcopy** commands perform various checks prior to installing or copying the *software_selections*, for example disk space analysis.

Options

swinstall and **swcopy** support the following options:

XToolKit Options
: The **swinstall** and **swcopy** commands support a subset of the standard X Toolkit options to control the appearance of the GUI. The supported options are: **-bg**, **-background**, **-fg**, **-foreground**, **-display**, **-name**, **-xrm**, and **-synchronous**. See the *X*(1) manual page by typing **man X** for a definition of these options.

-i
: Runs the command in interactive mode (Graphical User Interface). See the **Interactive Operation** and **Remote Operation** headings above for details.

-l
: (*Applies only to* **HP-UX 10.X**.) Runs the command in *linkinstall* mode which makes software installed under a server's *shared root* available to a diskless client's *private root* (HP-UX only).

  When run in the *linkinstall* mode, **swinstall**:

  - Creates NFS mounts to the software to make it accessible from the target. This may involve delayed mounting for alternate roots.
  - Modifies the target's *fstab* file.
  - Modifies the source's **exports** file to add mount permission for the target.

  Mounts are created by examining the *share_link* product attribute. Not all products support **linkinstall**. Some products may be visible without creating a new mount if they reside under an old one.

-p
: Previews an install task by running the session through the analysis phase only.

-r
: Causes **swinstall** to operate on alternate root directories, which must be specified the **@** *target_selections* option. Configuration scripts are not run on alternate roots. (This option is not required for alternate root operations but is maintained for backward compatibility. See the **Alternate Root Directory and Depot Directory** heading in *sd*(5) for more information.)

-v
: Turns on verbose output to stdout. (The **swinstall** or **swcopy** logfile is not affected by this option.) Verbose output is enabled by default; see the **verbose** option below.

-c *catalog*
: Specifies the pathname of an exported catalog which stores copies of the response file or files created by a request script (if **-x ask=true** or **-x ask=as_needed**). The response files are also stored in the **Installed Products Database** after the installation process is complete.

-C *session_file*
: Save the current options and operands to *session_file*. You can enter a relative or absolute path with the file name. The default directory for session files is **$HOME/.sw/sessions/**. You can recall a session file with the **-S** option.

-f *software_file*
: Read the list of *software_selections* from *software_file* instead of (or in addition to) the command line.

-J *jobid*
: Executes the previously scheduled job. This is the syntax used by the daemon to start the job.

-Q *date*
: Schedules the job for this date. You can change the date format by modifying the **/var/adm/sw/getdate.templ** file.

-s *source*
: Specifies the source depot (or tape) from which software is installed or copied. (SD can read both **tar** and **cpio** tape depots.) The default source type is *directory*. The syntax is:

S

[*host*][**:**][**/***directory*]

A host may be specified by its host name, domain name, or Internet address. A directory must be specified by an absolute path.

-S *session_file*
Execute **swinstall** or **swcopy** based on the options and operands saved from a previous session, as defined in *session_file*. You can save session information from a command-line session with the **-C** *session_file* option.

-t *target_file* Read the list of *target_selections* from *target_file* instead of (or in addition to) the command line.

-x *option=value*
Set the session *option* to *value* and override the default value (or a value in an alternate *option_file* specified with the **-X** option). Multiple **-x** options can be specified.

-X *option_file* Read the session options and behaviors from *option_file*.

Operands

The **swinstall** and **swcopy** commands support two types of operands: *software selections* followed by *target selections*. These operands are separated by the "@" (at) character. This syntax implies that the command operates on "software selections at targets".

Software Selections

The *selections* operands consist of *software_selections*.

swinstall and **swcopy** support the following syntax for each *software_selection*:

bundle[**.***product*[**.***subproduct*][**.***fileset*]][**,***version*]

product[**.***subproduct*][**.***fileset*][**,***version*]

- The **=** (equals) relational operator lets you specify selections with the following shell wildcard and pattern-matching notations:

  [ ], *****, **?**

  For example, the following expression installs all bundles and products with tags that end with "man":

  ```
  swinstall -s sw_server *man
  ```

- *Bundles* and *subproducts* are recursive. *Bundles* can contain other *bundles* and *subproducts* can contain other *subproducts*. For example:

  ```
  swinstall bun1.bun2.prod.sub1.sub2.fset,r=1.0
  ```

  or (using expressions):

  ```
  swinstall bun[12].bun?.prod.sub*,a=HP-UX
  ```

- The **\*** software specification selects all products. Use this specification with caution.

The **version** component has the form:

[**,r** <*op*> *revision*][**,a** <*op*> *arch*][**,v** <*op*> *vendor*]
[**,c** <*op*> *category*][**,q=***qualifier*][**,l=***location*]
[**,fr** <*op*> *revision*][**,fa** <*op*> *arch*]

- *location* applies only to installed software and refers to software installed to a location other than the default product directory.
- **fr** and **fa** apply only to filesets.
- The <*op*> (relational operator) component can be of the form:

  =, **==**, **>=**, **<=**, **<**, **>**, or **!=**

  which performs individual comparisons on dot-separated fields.

  For example, **r>=B.10.00** chooses all revisions greater than or equal to **B.10.00**. The system compares each dot-separated field to find matches.

S

- The **=** (equals) relational operator lets you specify selections with the shell wildcard and pattern-matching notations:

  [ ], *****, **?**, **!**

  For example, the expression **r=1[01].*** returns any revision in version 10 or version 11.
- All version components are repeatable within a single specification (e.g. **r>=A.12**, **r<A.20**). If multiple components are used, the selection must match all components.
- **Fully qualified software specs** include the **r=**, **a=**, and **v=** version components even if they contain empty strings. For installed software, **l=** is also included.
- No space or tab characters are allowed in a software selection.
- The software *instance_id* can take the place of the version component. It has the form:

  [*instance_id*]

  within the context of an exported catalog, where *instance_id* is an integer that distinguishes versions of products and bundles with the same tag.

The **\*** software specification selects all products. It is not allowed when removing software from the root directory **/**.

Target Selection

The **swinstall** and **swcopy** commands support the following syntax for each *target_selection*. The **:** (colon) is required if both a host and directory are specified.

[*host*][**:**][**/***directory*]

A host may be specified by its host name, domain name, or Internet address. A directory must be specified by an absolute path.

For *linkinstall*, on HP-UX 10.X systems: if the [*directory*] part of the selection is a relative path, then the value of *default.shared_root=true* is pre-pended for sources and the value of *default.private_root=true* is pre-pended for targets. These are normally **/export/shared_roots** and **/export/private_roots**, respectively.

EXTERNAL INFLUENCES

Default Options

In addition to the standard options, several SD behaviors and policy options can be changed by editing the default values found in:

/var/adm/sw/defaults	the system-wide default values.
$HOME/.swdefaults	the user-specific default values.

Values must be specified in the defaults file using this syntax:

[*command_name***.**]*option***=***value*

The optional *command_name* prefix denotes one of the SD commands. Using the prefix limits the change in the default value to that command. If you leave the prefix off, the change applies to all commands.

You can also override default values from the command line with the **-x** or **-X** options:

command **-x** *option***=***value*

command **-X** *option_file*

The following section lists all of the keywords supported by the **swinstall** and **swcopy** commands. If a default value exists, it is listed after the "=".

admin_directory=/var/adm/sw (for normal mode)
admin_directory=/var/home/LOGNAME/sw (for nonprivileged mode)

The location for SD logfiles and the default parent directory for the installed software catalog. The default value is **/var/adm/sw** for normal SD operations. When SD operates in nonprivileged mode (that is, when the **run_as_superuser** default option is set to **true**):

- The default value is forced to **/var/home/LOGNAME/sw**.
- The path element **LOGNAME** is replaced with the name of the invoking user, which SD reads from the system password file.

S

- If you set the value of this option to **HOME/***path*, SD replaces **HOME** with the invoking user's home directory (from the system password file) and resolves *path* relative to that directory. For example, **HOME/my_admin** resolves to the **my_admin** directory in your home directory.
- If you set the value of the **installed_software_catalog** default option to a relative path, that path is resolved relative to the value of this option.

SD's nonprivileged mode is intended only for managing applications that are specially designed and packaged. This mode cannot be used to manage the HP-UX operating system or patches to it. For a full explanation of nonprivileged SD, see the *Software Distributor Administration Guide*, available at the **http://docs.hp.com** web site.

See also the **installed_software_catalog** and **run_as_superuser** options.

agent_auto_exit=true
Causes the target agent to automatically exit after Execute phase, or after a failed Analysis phase. This is forced to **false** when the controller is using an interactive UI, or when -p (preview) is used. This enhances network reliability and performance. The default is **true** - the target agent automatically exits when appropriate. If set to **false**, the target agent will not exit until the controller ends the session.

agent_timeout_minutes=10000
Causes a target agent to exit if it has been inactive for the specified time. This can be used to make target agents more quickly detect lost network connections since RPC can take as long as 130 minutes to detect a lost connection. The recommended value is the longest period of inactivity expected in your environment. For command line invocation, a value between 10 minutes and 60 minutes is suitable. A value of 60 minutes or more is recommended when the GUI is used. The default of 10000 is slightly less than 7 days.

allow_downdate=false
(*Applies only to* **swinstall**.) Prevents the installation of an older revision of fileset that already exists at the target(s). (Many software products do not support "downdating".) If set to **true**, the older revision can be installed.

allow_incompatible=false
(*Applies only to* **swinstall**.) Requires that the software products which are being installed be "compatible" with the target selections. (All of the target selections must match the list of supported systems defined for each selected product.) If set to **true**, target compatibility is not enforced.

allow_multiple_versions=false
(*Applies only to* **swinstall**.) Prevents the installation of another, independent version of a product when a version already is already installed at the target.

If set to **true**, another version of an existing product can be installed into a new location. Multiple versions can only be installed if a product is locatable. Multiple configured versions will not work unless the product supports it.

S

allow_split_patches=false
Permits the use of single patch filesets without "sibling" filesets. In the default state of **false**, installation or copying of a single fileset from a multi-fileset patch automatically includes any other fileset that are part of the patch, based on the ancestor filesets of the target fileset. (This behavior applies to filesets selected directly by the user and to filesets automatically selected by SD to resolve software dependencies.)

When set to **true**, SD allows a single patch fileset to be installed or copied without including the sibling filesets. This allows a target to contain a patch that has been "split" into its component filesets. WARNING: Splitting a patch can create a situation in which one fileset in a sibling group would be updated by a patch, while the other filesets would remain at an earlier release.

ask=false
(*Applies only to* **swinstall**.) When **ask=true**, executes a request script which asks for a user response. If **ask=as_needed**, the **swinstall** command first determines if a response file already exists in the catalog specified in the **-c** option or source depot and executes the request script only when a response file is absent.

If set to **ask=true**, or **ask=as_needed**, you can use the **-c** *catalog* option to specify the pathname of an exported catalog to store copies of the response file or files created by the request script.

See *swask*(1M) for more information on request scripts.

autoreboot=false

(*Applies only to* **swinstall**.) Prevents the installation of software requiring a reboot from the non-interactive interface. If set to **true**, this software can be installed and the target system(s) will be automatically rebooted.

An interactive session always asks for confirmation before software requiring a reboot is installed.

autorecover=false

This option permits automatic recovery of original filesets if an installation error occurs. The cost is a temporary increase in disk space and slower performance. The default value of **false** causes **swinstall** to remove the original files as a fileset is updated. If an error occurs during the installation (e.g. network failure), then the original files are lost, and you must reinstall the fileset.

If set to **true**, all files are saved as backup copies until the current fileset finishes loading. If an error occurs during installation, the fileset's original files are replaced, and **swinstall** continues to the next fileset in the product or the product **postinstall** script.

When set to **true**, this option also affects scripts. For example, if a *preinstall* script fails, this option causes the corresponding *unpreinstall* script to execute. See *Software Distributor Administration Guide* for complete information.

autorecover_product=false

(*Applies only to* **swinstall**.) This option permits automatic recovery of original product files if an installation error occurs. The cost is a temporary increase in disk space and slower performance. The default value of **false** causes **swinstall** to remove any existing product files as a product is updated. If an error occurs during installation (e.g. network failure), then the original files are lost, and you must reinstall the product.

If set to **true**, all files for a product are saved as backup copies until the entire product finishes loading. Then the files are removed. If an error occurs during installation, the original product files are replaced, and **swinstall** exits.

When set to **true**, this option also affects scripts. For example, if a *preinstall* script fails, this option causes the corresponding *unpreinstall* script to execute. See *Software Distributor Administration Guide* for complete information.

autoremove_job=false

Controls automatic job removal of completed jobs. If the job is automatically removed, job information (job status or target logfiles) cannot be queried with **swjob**.

autoselect_dependencies=true

Automatically select dependencies when software is being selected. When set to **true**, and any software which has dependencies is selected for install, **swinstall** or **swcopy** makes sure that the dependencies are met. If they are not already met, they are automatically selected for you. If set to **false**, automatic selections are not made to resolve requisites. When set to **as_needed**, autoselected dependencies are operated only if the dependency is not already met on the target.

autoselect_patches=true

Automatically selects the latest patches (based on superseding and ancestor attributes) for a software object that a user selects for a **swinstall** or **swcopy** operation. When set to **false**, the patches corresponding to the selected object, are not automatically selected.

The **patch_filter=** option can be used in conjunction with **autoselect_patches**.

autoselect_reference_bundles=true

If **true**, bundles that are **sticky** are automatically installed or copied, along with the software it is made up of. If **false**, the software can be installed, or copied, without automatically including sticky bundles that contain it.

codeword=

Provides the "codeword" needed to unlock protected HP CD-ROM software.

Some HP software products are shipped on CD-ROM as "protected" products. That is, they cannot be installed or copied unless a "codeword" and "customer ID" are provided. The codeword is found on the CD-ROM certificate which you received from HP. You may use this default specification on the command line or the SD-UX Interactive User Interface to enter the codeword.

This default stores the codeword for future reference, and you need to enter the codeword only once. If you purchase a new HP product and a previous codeword has already been entered for that CD-ROM, just enter the new codeword as usual and the codewords will be merged internally.

NOTE: For HP-UX B.10.10 and later systems, SD searches the **.codewords** file on the server that is providing protected software to other hosts. It looks for valid customer_id/codeword pairs. In doing so, SD eliminates the need to enter codewords and customer_ids on every host that is "pulling" the software.

To properly store the customer_id/codeword for a CD-ROM, run **swinstall -p** or **swcopy -p** on the host serving the CD-ROM. After the codeword has been stored, clients installing or copying software using that host and CD-ROM as a source will no longer need a codeword or customer_id.

compress_files=false

If set to **true**, uncompressed files are compressed before transfer from a source. This enhances performance on slower networks for **swcopy** and **swinstall**, and results in smaller depots for **swcopy** and **swpackage**, unless the **uncompress_files** option is also set to **true**.

compress_index=false

Determines whether SD commands create compressed INDEX and INFO catalog files when writing to target depots or roots. The default of **false** does not create compressed files. When set to **true**, SD creates compressed and uncompressed INDEX and INFO files. The compressed files are named INDEX.gz and INFO.gz, and reside in the same directories as the uncompressed files.

Compressed files can enhance performance on slower networks, although they may increase disk space usage due to a larger Installed Products Database and depot catalog. SD controllers and target agents for HP-UX 11.01 and higher automatically load the compressed INDEX and INFO files from the source agent when:

- The source agent supports this feature.
- INDEX.gz or INFO.gz exist on the source depot.
- INDEX.gz or INFO.gz are not older than the corresponding uncompressed INDEX or INFO files.

The uncompressed INDEX or INFO file is accessed by the source agent if any problem occurs when accessing, transferring, or uncompressing the INDEX.gz or INFO.gz file.

controller_source=

Specifies the location of a depot for the controller to access to resolve selections. Setting this option can reduce network traffic between the controller and the target. Use the target selection syntax to specify the location: [*host*][**:**][**/***directory*]

The **controller_source_option** supports the same syntax as the **-s** *source* option. This option has no effect on which sources the target uses and is ignored when used with the Interactive User Interface.

create_target_path=true

Causes the agent to create the target directory if it does not already exist. If set to **false**, a new target directory is not created. This option can prevent the erroneous creation of new target depots or new alternate root directories.

create_time_filter=0

For cumulative source depots, this option allows consistent software selections over time by **swlist**, **swcopy**, and **swinstall**. The default of zero includes all bundles, products, subproducts, and filesets in the source depot as candidates for selection (and autoselection of dependencies and patches), based on the software selections and other options. When set to a time (specified as seconds from epoch), only those bundles, products, and filesets (and

S

the subproducts in the product) with a create_time less than or equal to the specified value are available for selection (or autoselection). To list the create_time of bundles, products and filesets, use:

swlist -a create_time -a create_date

customer_id=
This number, also printed on the Software Certificate, is used to "unlock" protected software and restrict its installation to a specific site or owner. It is entered using the **-x** *customer_id=* option or by using the Interactive User Interface. The *customer_id* can be used on any HP-UX 10.0X or later system.

defer_configure=false
(*Applies only to* **swinstall**.) Causes **swinstall** to automatically run configure scripts for the *software_selections* after they are installed. (Alternate root directories are not configured.)

When set to true, **swinstall** does not run configure scripts. If you want to configure the software later, you must run the **swconfig** command.

NOTES:

- Multiple versions of a product will not be automatically configured if another version is already configured. Use the **swconfig** command to configure multiple versions separately.
- SD ignores this option when it installs software that causes a system reboot.

distribution_source_directory=/var/spool/sw
Defines the default location of the source depot (when the **source_type** is *directory*). You can also use the *host*:*path* syntax. The **-s** option overrides this default.

distribution_target_directory=/var/spool/sw
(*Applies only to* **swcopy**.) Defines the default location of the target depot.

enforce_dependencies=true
Requires that all dependencies specified by the *software_selections* be resolved either in the specified source, or at the *target_selections* themselves.

The **swinstall** and **swcopy** commands will not proceed unless the dependencies have also been selected or already exist at the target in the correct state (INSTALLED or AVAILABLE). This prevents unusable software from being installed on the system. It also ensures that depots contain usable sets of software.

If set to **false**, dependencies are still checked, but not enforced. Corequisite dependencies, if not enforced, may keep the selected software from working properly. Prerequisite dependencies, if not enforced, may cause the installation or configuration to fail.

S

enforce_dsa=true
Prevents the command from proceeding past the analysis phase if the disk space required is beyond the available free space of the impacted filesystem(s). If set to **false**, the install or copy operation uses the filesystem's minfree space and may fail because it reaches the filesystem's absolute limit.

enforce_kernbld_failure=true
(*Applies only to* **swinstall**.) Prevents **swinstall** from proceeding past the kernel build phase if the kernel build processes fail. If set to **false**, the install operation continues (without suspension if in the interactive mode) despite failure or warnings from either the system preparation process or the kernel build process.

When set to the default value of **true**, this option generates an error if a command tries to relocate a non-relocatable fileset. (Relocatable filesets are packaged with the **is_relocatable** attribute set to **true**). When set to **false**, the usual error handling process is overridden, and SD permits the command to relocate the fileset.

enforce_scripts=true
Controls the handling of errors generated by scripts. If **true**, and a script returns an error, an error message appears reporting that the execution phase failed. If **false**, **swinstall** attempts to continue operation. A warning message appears saying that the analysis or execution phase succeeded. The message identifies the specific **swinstall**

phase (checkinstall, preinstall, postinstall, or configure).

fix_explicit_directories=false

Controls the **swinstall** response to explicitly packaged software (software packaged with explicit file specifications). The default value of **false** causes **swinstall** to set permissions (as specified in the product specification file) on new directories but never on pre-existing directories. When set to **true**, **swinstall** also sets the permissions on pre-existing directories.

installed_software_catalog=products

(*Applies only to* **swinstall**.) Defines the directory path where the Installed Products Database (IPD) is stored. This information describes installed software. When set to an absolute path, this option defines the location of the IPD. When this option contains a relative path, the SD controller appends the value to the value specified by the **admin_directory** option to determine the path to the IPD. For alternate roots, this path is resolved relative to the location of the alternate root. This option does not affect where software is installed, only the IPD location.

This option permits the simultaneous installation and removal of multiple software applications by multiple users or multiple processes, with each application or group of applications using a different IPD.

Caution: use a specific **installed_software_catalog** to manage a specific application. SD does not support multiple descriptions of the same application in multiple IPDs.

See also the **admin_directory** and **run_as_superuser** options, which control SD's nonprivileged mode. (This mode is intended only for managing applications that are specially designed and packaged. This mode cannot be used to manage the HP-UX operating system or patches to it. For a full explanation of nonprivileged SD, see the *Software Distributor Administration Guide*, available at the **http://docs.hp.com** web site.)

job_title=

This is an ASCII string giving a title to a job. It is displayed along with the job ID to provide additional identifying information about a job when **swjob** or **sd** is invoked. The default value is to have no title. If a title is specified, it should be enclosed in quotes.

layout_version=1.0

(Applies only to **swcopy**.*)* Specifies the POSIX **layout_version** to which the SD commands conform when writing distributions and **swlist** output. Supported values are "1.0" (default) and "0.8".

SD object and attribute syntax conforms to the *layout_version 1.0* specification of the *IEEE POSIX 1387.2 Software Administration* standard. SD commands still accept the keyword names associated with the older layout version, but you should use **layout_version=0.8** only to create distributions readable by older versions of SD.

See the description of the **layout_version** option in *sd*(5) for more information.

logdetail=false

Controls the amount of detail written to the logfile. When set to **true**, this option adds detailed task information (such as options specified, progress statements and additional summary information) to the logfile. This information is in addition to log information controlled by the **loglevel** option.

See **loglevel=1** and the *sd*(5) manual page by typing **man 5 sd** for more information.

logfile=/var/adm/sw/swremove.log

This is the default command log file for the **swinstall** command.

loglevel=1

Controls the log level for the events logged to the command logfile, the target agent logfile, and the source agent logfile. This information is in addition to the detail controlled by the **logdetail** option. (See **logdetail=false** and the *sd*(5) manual page for more information.) A value of:

0 provides no information to the logfile.
1 enables verbose logging to the logfiles.
2 enables very verbose logging, including per-file messages, to the logfiles.

S

log_msgid=0
Adds numeric identification numbers at the beginning of SD logfile messages:
0 (default) No identifiers are attached to messages.
1 Adds identifiers to ERROR messages only.
2 Adds identifiers to ERROR and WARNING messages.
3 Adds identifiers to ERROR, WARNING, and NOTE messages.
4 Adds identifiers to ERROR, WARNING, NOTE, and certain other informational messages.

match_target=false
(*Applies only to* **swinstall**.) If set to **true**, software selection is done by locating filesets on the source that match the target system's installed filesets. If multiple targets are specified, the first in the list is used as the basis for selections.

max_targets=25
When set to a positive integer, SD limits the number of concurrent install or copy operations to the number specified. As each copy or install operation completes, another target is selected and started until all targets have been completed.

Server and network performance determines the optimal setting; a recommended starting point is 25 (the default value). If you set this option to a value of less than one, SD attempts to install or copy to all targets at once.

mount_all_filesystems=true
Attempt to mount all filesystems in the **/etc/fstab** file at the beginning of the analysis phase, to ensure that all listed filesystems are mounted before proceeding. This policy helps to ensure that files are not loaded into a directory that may be below a future mount point.

If set to **false**, the mount operation is not attempted, and no check of the current mounts is performed.

os_name
(*Applies only to* **swinstall**.) This option can be used in conjunction with **os_release** to specify the desired OS name during an HP-UX update. The **os_name** option should only be specified from the command line. Refer to the SD **readme** file for correct syntax. You can display the **readme** file by entering:

```
swlist -a readme -l product SW-DIST
```

os_release
(*Applies only to* **swinstall**.) This option can be used in conjunction with **os_name** to specify the desired OS release during an HP-UX update. The **os_release** option should only be specified from the command line. Refer to the SD **readme** file for correct syntax. You can display the **readme** file by entering:

```
swlist -a readme -l product SW-DIST
```

S

patch_filter=software_specification
This option can be used in conjunction with the **autoselect_patches** or **patch_match_target** options to filter the selected patches to meet the criteria specified by *software_specification*. The default value of this option is ***.***.

patch_match_target=false
If set to **true**, this option selects the latest patches (software identified by the *is_patch=true* attribute) that correspond to software on the target root or depot.

The **patch_filter=** option can be used in conjunction with **patch_match_target**.

patch_save_files=true
(Applies only to **swinstall***)* Saves the original versions of files modified by patches, which permits the future rollback of a patch. Patched files are saved to **/var/adm/sw/save**. When set to **false**, patches cannot be rolled back (removed) unless the base software modified by the patch is removed at the same time.

To commit a patch by removing the corresponding saved files, use the **swmodify** command's **patch_commit** option.

polling_interval=2
Defines the polling interval, in seconds, used by the interactive GUI or TUI of the

controller. It specifies how often each target agent is polled to obtain status information about the task being performed. When operating across wide-area networks, the polling interval can be increased to reduce network overhead.

preserve_create_time=false
(*Applies only to* **swcopy**.) Preserves the original create time when you copy depots, which produces consistent results when you use the copies. The default of **false** sets the create_time of software bundles, products, and filesets equal to the time the object was created in the depot. When set to **true**, the create_time of software bundles, products, and filesets is set to that specified in the source depot. Note that using this option when copying to a master depot can change the objects that are visible when you use the **create_time_filter** option.

recopy=false
(*Applies only to* **swcopy**.) Do not copy a fileset that is already available on the target at the same version. If **recopy=true**, copy the fileset in any case.

register_new_depot=true
(*Applies only to* **swcopy**.) Causes **swcopy** to register a newly created depot with the local **swagentd**. This action allows other SD commands to automatically "see" this depot. If set to **false**, a new depot is not automatically registered. It can be registered later with the **swreg** command.

register_new_root=true
(*Applies only to* **swinstall**.) Causes alternate roots to be registered during **swinstall**. These can be listed with **swlist**.

reinstall=false
When re-installing an existing revision of a fileset, this option causes that fileset to be skipped, i.e. not re-installed. If set to **true**, the fileset is re-installed. See also **recopy=false**.

reinstall_files=false
Controls the overwriting of files, which may enhance performance on slow networks or disks. At the default value of false, SD compares each file in a source fileset to corresponding files on the target system. SD compares the files based on size, timestamp, and (optionally) the *checksum* (see **reinstall_files_use_cksum**). If the files are identical the files on the target system are not overwritten.

When set to true, SD does not compare files and overwrites any identical files on the target.

reinstall_files_use_cksum=true
Controls the use of checksum comparisons when the **reinstall_files** option is set to false. At the default value of true, this option causes SD to compute and compare checksums to determine if a new file should overwrite an old file. Use of checksums slows the comparison but is a more robust check for equivalency than size and time stamp.

If set to false, SD does not compute checksums and compares files only by size and timestamp.

remove_obsolete_filesets=false
(*Applies only to* **swcopy**.) Controls whether **swcopy** automatically removes obsolete filesets from target products in the target depot. If set to **true**, **swcopy** removes obsolete filesets from the target products that were written to during the copy process. Removal occurs after the copy is complete. Filesets are defined as obsolete if they were not part of the most recent packaging of the product residing on the source depot.

retry_rpc=1
Defines the number of times a lost source connection is retried during file transfers in **swinstall** or **swcopy**. A lost connection is one that has timed out. When used in conjunction with the **rpc_timeout** option, the success of installing over slow or busy networks can be increased. If set to zero, any **rpc_timeout** to the source causes the task to abort. If set from 1 to 9, the install of each fileset is attempted that number of times. The **reinstall_files** option should also be set to false to avoid installing files within the fileset that were successfully installed.

This option also applies to the controller contacting the agent. If the agent session fails to start for any reason, the controller tries to recontact that agent for the number of times specified in **retry_rpc**, using the values from the **retry_rpc_interval** option to determine how long to wait between each attempt to recontact the agent.

retry_rpc_interval={0}

Specifies in minutes the length of the interval for repeated attempts to make a connection to a target after an initial failure. Used in conjunction with the **retry_rpc** option. If the number of values in this option equals the value of **retry_rpc**, SD tries reestablishing a source connection for the number of times specified in **retry_rpc**. If the number of values in **retry_rpc_interval** is less than the value in **retry_rpc**, SD repeats the final interval value until the number of retries matches **retry_rpc**.

For example, if a session failed to start and **retry_rpc** was set to 9 and **retry_rpc_interval** was set to {1 2 4 8 15} to allow long waits to handle transient network failures, the SD controller would attempt to recontact the agent after 1 minute for the first retry, then 2 minutes for the second retry, 4 for the third, then 8, then 15 for all additional retries until nine retries were attempted. With these values, a file load failure could cause the operation to pause for 90 minutes (1+2+4+8+15+15+15+15+15). If **retry_rpc** was set to 5 and **retry_rpc_interval** was set to {1 2 4 8 15}, the controller would try to contact the target five times over a 30-minute period.

rpc_binding_info=ncacn_ip_tcp:[2121] ncadg_ip_udp:[2121]

Defines the protocol sequence(s) and endpoint(s) on which the daemon listens and the other commands contact the daemon. If the connection fails for one protocol sequence, the next is attempted. SD supports both the tcp (ncacn_ip_tcp:[2121]) and udp (ncadg_ip_udp:[2121]) protocol sequence on most platforms. See the *sd*(5) man page by typing **man 5 sd** for more information.

rpc_timeout=5.

Relative length of the communications timeout. This is a value in the range from 0 to 9 and is interpreted by the DCE RPC. Higher values mean longer times; you may need a higher value for a slow or busy network. Lower values give faster recognition on attempts to contact hosts that are not up or not running **swagentd**. Each value is approximately twice as long as the preceding value. A value of 5 is about 30 seconds for the **ncadg_ip_udp** protocol sequence. This option may not have any noticeable impact when using the **ncacn_ip_tcp** protocol sequence.

run_as_superuser=true

This option controls SD's nonprivileged mode. This option is ignored (treated as true) when the invoking user is super-user.

When set to the default value of true, SD operations are performed normally, with permissions for operations either granted to a local super-user or set by SD ACLs. (See *swacl*(1M) for details on ACLs.)

When set to false and the invoking user is local and is *not* super-user, nonprivileged mode is invoked:

- Permissions for operations are based on the user's file system permissions.
- SD ACLs are ignored.
- Files created by SD have the uid and gid of the invoking user, and the mode of created files is set according to the invoking user's umask.

SD's nonprivileged mode is intended only for managing applications that are specially designed and packaged. This mode cannot be used to manage the HP-UX operating system or patches to it. For a full explanation of nonprivileged SD, see the *Software Distributor Administration Guide*, available at the **http://docs.hp.com** web site.

See also the **admin_directory** and **installed_software_catalog** options.

select_local=true

If no *target_selections* are specified, select the default root directory **/** (**swinstall**), or the default **target_directory** (**swcopy**), at the local host as the target of the command.

software=

Defines the default *software_selections*. There is no supplied default. If there is more than one software selection, they must be separated by spaces.

software_view=all_bundles
Indicates the software view to be used as the default level for the software listing in the GUI. It can be set to **all_bundles**, **products**, or a bundle category tag (to indicate to show only bundles of that category).

source=
Specify a source to automatically bypass the GUI and CLI source selection dialog box. This has the same effect as the **-s***source* command line option. Specify the source using the following syntax.

[*path*]

source_cdrom=/SD_CDROM
Defines the default location of the source CD-ROM using the syntax [*host*]**:**[*path*].

source_tape=/dev/rmt/0m
Defines the default location of the source tape, usually the character-special file of a local tape device. You can also use the [*host*]**:**[*path*] syntax, but the host must match the local host. The **-s** option overrides this value. (Note that SD can read both **tar** and **cpio** tape depots.)

source_type=directory
Defines the default source type: **cdrom**, **directory**, or **tape**. The source type derived from the **-s** option overrides this value. (SD can read both **tar** and **cpio** tape depots.)

targets=
Defines the default *target_selections*. There is no supplied default (see **select_local** above). If there is more than one target selection, they must be separated by spaces.

uncompress_files=false
(*Applies only to* **swcopy**.) If set to **true**, files being transferred from a source are uncompressed before **swcopy** store them on the target depot.

use_alternate_source=false
Empowers each target agent to use its own, configured alternate source, instead of the one specified by the user. If **false**, each target agent uses the same source (the source specified by the user and validated by the command). If **true**, each target agent uses its own configured value for the source.

verbose=1
Controls the verbosity of the output (stdout). A value of

0	disables output to stdout. (Error and warning messages are always written to stderr).
1	enables verbose messaging to stdout.

write_remote_files=false
Prevents the installation or copying of files to a target which exists on a remote filesystem. All files destined for a remote filesystem are skipped.

If set to **true** and if the superuser has write permission on the remote filesystem, the remote files are installed or copied.

Session File

Each invocation of the **swinstall** or **swcopy** command defines an installation or copy session. The invocation options, source information, software selections, and target hosts are saved before the installation or copy task actually commences. This lets you re-execute the command even if the session ends before proper completion.

Each session is saved to the file **$HOME/.sw/sessions/swinstall{swcopy}.last**. This file is overwritten by each invocation of **swinstall** or **swcopy**.

You can also save session information from interactive or command-line sessions. From an interactive session, you can save session information into a file at any time by selecting the *Save Session* or *Save Session As* option from the *File* menu. From a command-line session, you can save session information by executing **swinstall** or **swcopy** with the **-C** *session__file* option.

A session file uses the same syntax as the defaults files. You can specify an absolute path for a session file. If you do not specify a directory, the default location for a session file is **$HOME/.sw/sessions/**.

To re-execute a saved session from an interactive session, use the *Recall Session* option from the *File* menu. To re-execute a session from a command-line, specify the session file as the argument for the **-S** *session__file* option of **swinstall** or **swcopy**.

Note that when you re-execute a session file, the values in the session file take precedence over values in the system defaults file. Likewise, any command line options or parameters that you specify when you invoke **swinstall** or **swcopy** take precedence over the values in the session file.

Software and Target Lists

Most SD commands support software and target selections from separate input files (see the **-f** and **-t** command-line options). Software and targets specified in these files will be selected for operation. **swinstall** and **swcopy** also support an interactive read and save of target and software groups. Target and software groups can be saved in files (default location **$HOME/.sw/targets/** and **$HOME/.sw/software/**) and then selected in subsequent **swinstall** and **swcopy** operations.

Additionally, the **swinstall** and **swcopy** interactive user interfaces read a default list of hosts on which to operate. The list is stored in:

/var/adm/sw/defaults.hosts	the system-wide default list of hosts
$HOME/.swdefaults.hosts	the user-specific default list of hosts

For each interactive command, target hosts containing roots and target hosts containing depots are specified in separate lists (**hosts**, **hosts_with_depots**, respectively). The list of hosts are enclosed in {} braces and separated by white space (blank, tab and newline). For example:

```
swinstall.hosts={hostA hostB hostC hostD hostE hostF}
swcopy.hosts_with_depots={hostS}
```

The **swinstall** and **swcopy** interactive user interfaces read a default list of patch filters that you can use as selection criteria for patch software. The list is stored in:

/var/adm/sw/defaults.patchfilters	the system-wide default list of patch filters.
$HOME/.sw/defaults.patchfilters	the user-specific default list of patch filters.

The list of patch filters is enclosed in braces {} and separated by white space (blank, tab, or newline). For example:

```
swinstall.patch_filter_choices={
*.*,c=enhancement
*.*,c=critical
}
swcopy.patch_filter_choices={
Product.Fileset,c=halts_system
}
```

Environment Variables

The environment variable that affects the **swinstall** command is:

LANG Determines the language in which messages are displayed. If LANG is not specified or is set to the empty string, a default value of **C** is used. See the *lang*(5) man page by typing **man 5 sd** for more information.

NOTE: The language in which the SD agent and daemon log messages are displayed is set by the system configuration variable script, **/etc/rc.config.d/LANG**. For example, **/etc/rc.config.d/LANG**, must be set to **LANG=ja_JP.SJIS** or **LANG=ja_JP.eucJP** to make the agent and daemon log messages display in Japanese.

LC_ALL Determines the locale to be used to override any values for locale categories specified by the settings of **LANG** or any environment variables beginning with **LC_**.

LC_CTYPE
Determines the interpretation of sequences of bytes of text data as characters (e.g., single- versus multibyte characters in values for vendor-defined attributes).

LC_MESSAGES
Determines the language in which messages should be written.

LC_TIME
Determines the format of dates (*create_date* and *mod_date*) when displayed by **swlist**.

Used by all utilities when displaying dates and times in **stdout**, **stderr**, and **logging**.

TZ Determines the time zone for use when displaying dates and times.

Environment variables that affect scripts:

SW_CATALOG
Holds the path to the Installed Products Database (IPD), relative to the path in the **SW_ROOT_DIRECTORY** environment variable. Note that you can specify a path for the IPD using the **installed_software_catalog** default option.

SW_CONTROL_DIRECTORY
Defines the current directory of the script being executed, either a temporary catalog directory, or a directory within in the Installed Products Database (IPD). This variable tells scripts where other **control** scripts for the software are located (e.g. subscripts).

SW_CONTROL_TAG
Holds the tag name of the control_file being executed. When packaging software, you can define a physical name and path for a control file in a depot. This lets you define the control_file with a name other than its tag and lets you use multiple control file definitions to point to the same file. A control_file can query the

SW_LOCATION
Defines the location of the product, which may have been changed from the default product directory. When combined with the **SW_ROOT_DIRECTORY**, this variable tells scripts where the product files are located.

SW_PATH
A PATH variable which defines a minimum set of commands available to for use in a **control** script (e.g. **/sbin:/usr/bin**).

SW_ROOT_DIRECTORY
Defines the root directory in which the session is operating, either "/" or an alternate root directory. This variable tells **control** scripts the root directory in which the products are installed. A script must use this directory as a prefix to **SW_LOCATION** to locate the product's installed files. The configure script is only run when **SW_ROOT_DIRECTORY** is **/**.

SW_SESSION_OPTIONS
Contains the pathname of a file containing the value of every option for a particular command, including software and target selections. This lets scripts retrieve any command options and values other than the ones provided explicitly by other environment variables. For example, when the file pointed to by **SW_SESSIONS_OPTIONS** is made available to a request script, the *targets* option contains a list of *software_collection_specs* for all targets specified for the command. When the file pointed to by **SW_SESSIONS_OPTIONS** is made available to other scripts, the *targets* option contains the single *software_collection_spec* for the targets on which the script is being executed.

SW_SOFTWARE_SPEC
This variable contains the fully qualified software specification of the current product or fileset. The software specification allows the product or fileset to be uniquely identified.

Additional environment variables that affect scripts for **swinstall**:

SW_DEFERRED_KERNBLD
This variable is normally unset. If it is set, the actions necessary for preparing the system file **/stand/system** cannot be accomplished from within the *postinstall* scripts, but instead must be accomplished by the *configure*scripts. This occurs whenever software is installed to a directory other than /, such as for a cluster client system. This variable should be read only by the *configure* and *postinstall* scripts of a kernel fileset. The **swinstall** command sets these environment variables for use by the kernel preparation and build scripts.

SW_INITIAL_INSTALL
This variable is normally unset. If it is set, the **swinstall** session is being run as the back end of an initial system software installation ("cold" install).

SW_KERNEL_PATH
The path to the kernel. The default value is **/stand/vmunix**, defined by the **swagent**

S

option or **kernel_path**.

SW_SESSION_IS_KERNEL
Indicates whether a kernel build is scheduled for the current install/remove session. A **TRUE** value indicates that the selected kernel fileset is scheduled for a kernel build and that changes to **/stand/system** are required. A null value indicates that a kernel build is not scheduled and that changes to **/stand/system** are not required.

The value of this variable is always equal to the value of **SW_SESSION_IS_REBOOT**.

SW_SESSION_IS_REBOOT
Indicates whether a reboot is scheduled for a fileset selected for removal. Because all HP-UX kernel filesets are also reboot filesets, the values of this variables is always equal to the value of **SW_SESSION_IS_KERNEL**.

SW_SYSTEM_FILE_PATH
The path to the kernel's system file. The default value is **/stand/system**.

Signals

The **swinstall** and **swcopy** commands catch the signals SIGQUIT, SIGINT, and SIGUSR1. If these signals are received, the command prints a message, sends a Remote Procedure Call (RPC) to the agents to wrap up after completion, and then exits.

The agent ignores SIGHUP, SIGINT, and SIGQUIT. It immediately exits gracefully after receiving SIGTERM, SIGUSR1, or SIGUSR2. Killing the agent may leave corrupt software on the system, and thus should only be done if absolutely necessary. Note that when an SD command is killed, the agent does not terminate until completing the task in progress.

The daemon ignores SIGHUP, SIGINT and SIGQUIT. It immediately exits gracefully after receiving SIGTERM and SIGUSR2. After receiving SIGUSR1, it waits for completion of a copy or remove from a depot session before exiting, so that it can register or unregister depots if necessary. Requests to start new sessions are refused during this wait.

Locking

SD commands use a common locking mechanism for reading and modifying the Installed Products Database (IPD) and software depots. This mechanism allows multiple readers but only one writer on an IPD or depot:

Write Locks

swinstall commands that modify the IPD are restricted from simultaneous modification using *fcntl*(2) locking on the file <*IPD location*>**/swlock** (e.g. **/var/adm/sw/products/swlock**).

swcopy commands that modify a software depot are restricted from simultaneous modification using *fcntl*(2) locking on the file <*depot directory*>**/catalog/swlock** (e.g. **/var/spool/sw/catalog/swlock**).

Read Locks

Both **swinstall** and **swcopy** commands set *fcntl*(2) read locks on source depots using the **swlock** file mentioned above. When a read lock is set, it prevents all SD commands from performing modifications (i.e. from setting write locks).

Terminal Support

For in-depth information about terminal support refer to:

- The *Software Distributor Administration Guide* manual
- Start the GUI or TUI, select the *Help* menu, then select the *Keyboard...* option to access the *Keyboard Reference Guide*.

RETURN VALUES

An interactive **swinstall** or **swcopy** session always returns 0. A non-interactive **swinstall** or **swcopy** session returns:

0 The *software_selections* were successfully installed/copied.
1 The install/copy operation failed on all *target_selections*.
2 The install/copy operation failed on some *target_selections*.

DIAGNOSTICS

The **swinstall** and **swcopy** commands write to stdout, stderr, and to specific logfiles.

Standard Output

An interactive **swinstall** or **swcopy** session does not write to stdout. A non-interactive **swinstall** or **swcopy** session writes messages for significant events. These include:

- a begin and end session message,
- selection, analysis, and execution task messages for each *target_selection*.

Standard Error

An interactive **swinstall** or **swcopy** session does not write to stderr. A non-interactive **swinstall** or **swcopy** session writes messages for all WARNING and ERROR conditions to stderr.

Logging

Both interactive and non-interactive **swinstall** and **swcopy** sessions log summary events at the host where the command was invoked. They log detailed events to the **swagent** logfile associated with each *target_selection*.

Command Log

The **swinstall** and **swcopy** commands log all stdout and stderr messages to the the logfile **/var/adm/sw/swinstall.log** (**/var/adm/sw/swcopy.log**). Similar messages are logged by an interactive **swinstall** and **swcopy** session. The user can specify a different logfile by modifying the **logfile** option.

Target Log

A **swagent** process performs the actual install or copy operation at each *target_selection*. For install tasks, the **swagent** logs messages to the file **var/adm/sw/swagent.log** beneath the root directory (e.g. **/** or an alternate root directory). For copy tasks, the **swagent** logs messages to the file **swagent.log** beneath the depot directory (e.g. **/var/spool/sw**).

You can view the command and target log files with the **swjob** or **sd** command.

Source Depot Audit Log

If both source and target machine are updated to SD revision B.11.00 or later, the system administrator at the source depot machine can track *which* user pulls *which* software from a depot on the source machine and *when* the software is pulled. (Note that a user running **swinstall/swcopy** from a target machine cannot set this option; only the administrator of the source depot machine can set it. See the *source_depot_audit* option in the *swagent*(1M) man page.)

EXAMPLES

swinstall

To invoke an interactive session of **swinstall**:

```
swinstall
```

Select the C and Pascal products from the network source software server (sw_server) and start an interactive session:

```
swinstall -i -s sw_server cc pascal
```

Install the C and Pascal products to a set of remote hosts:

```
swinstall -s sw_server cc pascal @ hostA hostB hostC
```

Update the HP Omniback product from a CD-ROM mounted at /cd :

```
swinstall -s /cd/swmedia Omniback
```

Install an incompatible version of HP Omniback into the directory **/exports**:

```
swinstall -x allow_incompatible=true -s/products Omniback,a=arch \
     @ /exports
```

Install all products from the cartridge tape **/dev/rmt/0**:

```
swinstall -s /dev/rmt/0 \*
```

Reinstall the *software_selections* listed in the file **/tmp/install.products** on the hosts listed in the file **tmp/install.hosts:**

```
swinstall -x reinstall=true -f/tmp/install.products \
     -t/tmp/install.hosts
```

Execute **swinstall** interactively using the session file **/tmp/case.selections** as a basis:

```
swinstall -i -S /tmp/case.selections
```

Install all the software from local depot **/tmp/sample.depot.1**, using any response files generated by request scripts:

```
swinstall -s /tmp/sample.depot.1 -x ask=true \*
```

Install **Product1** from remote depot **/tmp/sample.depot.1** on host **swposix** and use an existing response file (previously generated by the **swask** command) located in **/tmp/bar.depot**:

```
swinstall -s swposix:/tmp/sample.depot.1 -c /tmp/bar.depot Product1
```

Install all products in remote depot **/tmp/sample.depot.1** on host **swposix**, use any response files generated by request scripts, create catalog **/tmp/bar.depot** and copy all response files to the new catalog:

```
swinstall -s swposix:/tmp/sample.depot.1 -c /tmp/bar.depot \
     -x ask=true \*
```

Install all products in remote depot **/tmp/sample.depot.1 on host swposix**, use response files, run request scripts only when a response file is absent, create catalog **/tmp/bar.depot** and copy all response files to the new catalog:

```
swinstall -s swposix:/tmp/sample.depot.1 -c swposix:/tmp/bar.depot \
     -x ask=as_needed \*
```

Install all patches in the depot that correspond to currently installed software and are of the **critical** category:

```
swinstall -s /tmp/sample.depot.1 -x patch_match_target=true \
-x patch_filter=\"*.*, c=critical\"
```

The following example applies to HP-UX 10.X only.

To *linkinstall* the product TEST to the clients **clientA, clientB** from the server:

```
swinstall -l -r -s :OS_700 TEST @ clientA clientB
```

The following example applies to HP-UX 10.X only.

To *linkinstall* product TEST2 to your own "/" directory from an application server on "serve":

```
swinstall -l -s serve TEST2
```

swcopy

Invoke an interactive session of **swcopy**:

```
swcopy
```

Invoke an interactive session, using default depot at hostX as the source:

```
swcopy -i -s hostX
```

Copy all products from the cartridge tape **/dev/rmt/0m** to the default depot on the local host:

```
swcopy -s /dev/rmt/0m \*
```

Load the *software_selections* listed in the file **/tmp/load.products** using the default source/depot:

```
swcopy -f /tmp/load.products
```

Copy the C and Pascal products to some local and remote depots:

```
swcopy -s sw_server cc pascal @ /var/spool/sw hostA:/tmp/sw hostB
```

FILES

$HOME/.swdefaults
: Contains the user-specific default values for some or all SD options. If this file does not exist, SD looks for user-specific defaults in **$HOME/.swdefaults.hosts**.

$HOME/.sw/defaults.hosts
Contains the user-specific default list of hosts to manage.

$HOME/.sw/defaults.patchfilters
Contains the user-specific default list of patch filters.

$HOME/.sw/sessions/
Contains session files automatically saved by the SD commands or explicitly saved by the user.

/usr/lib/sw/sys.defaults
Contains the master list of current SD options with their default values.

/var/adm/sw/
The directory which contains all of the configurable and non-configurable data for SD. This directory is also the default location of logfiles.

/var/adm/sw/defaults
Contains the active system-wide default values for some or all SD options.

/var/adm/sw/defaults.hosts
Contains the system-wide default list of hosts to manage.

/var/adm/sw/defaults.patchfilters
Contains the system-wide default list of patch filters.

/var/adm/sw/getdate.templ
Contains the set of date/time templates used when scheduling jobs.

/var/adm/sw/products/
The Installed Products Database (IPD), a catalog of all products installed on a system.

/var/adm/sw/queue/
The directory which contains the information about all active and complete install jobs, copy jobs, and other jobs initiated by the SD commands.

/var/spool/sw/
The default location of a source and target software depot.

AUTHOR

swinstall and **swcopy** were developed by the Hewlett-Packard Company and Mark H. Colburn (see *pax*(1)).

SEE ALSO

swacl(1M), swagentd(1M), swask(1M), swconfig(1M), swjob(1M), swlist(1M), swmodify(1M), swpackage(1M), swreg(1M), swremove(1M), swverify(1M), install-sd(1M), update-ux(1M), sd(4), swpackage(4), sd(5).

Software Distributor Administration Guide, available at **http://docs.hp.com**.

SD customer web site at **http://software.hp.com/SD_AT_HP/**.

NAME

swjob, sd - display and monitor job information and create and remove jobs; invoke graphical user interface to display and monitor job information and create and remove jobs; respectively

SYNOPSIS

swjob [**-i**] [**-R**] [**-u**] [**-v**] [**-a** *attribute*] [**-C** *session_file*] [**-f** *jobid_file*] [**-S** *session_file*] [**-t** *target_file*] [**-x** *option=value*] [**-X** *option_file*] [*jobid(s)*] [**@** *target_selections*]

sd [*XToolkit Options*] [**-x** *option=value*] [**-X** *option_file*]

Remarks

- The **sd** command invokes an interactive interface to the same functionality that **swjob** provides. See **Interactive Operation** below for more details.
- This command supports operation on remote systems. See **Remote Operation** below for details.
- For an overview of all SD commands, see the *sd*(5) man page by typing **man 5 sd** on the command line.

DESCRIPTION

The **swjob** command displays job information and removes jobs. It supports these features:

- Display the current install jobs, copy jobs, and other SD jobs initiated by the SD commands.
- Specify a specific job to list or remove.
- Display the command logfile for a specific job.
- Display the target logfile for a specific target.

Remote Operation

You can enable Software Distributor (SD) to manage software on remote systems. To let the root user from a central SD *controller* (also called the *central management server* or *manager node*) perform operations on a remote *target* (also called the *host* or *agent*):

1) Install a special HP ServiceControl Manager fileset on the remote systems. This enables remote operations by automatically setting up the root, host, and template Access Control Lists (ACLs) on the remote machines and permitting root access from the controller system. To install the fileset, run the following command on each remote system:

swinstall -s *controller*:**/var/opt/mx/depot11 AgentConfig.SD-CONFIG**

NOTES:

- *controller* is the name of the central management server.
- If the target is running HP-UX 10.20, use the same command but substitute **depot10** for **depot11**.
- Targets previously set up by SD/OV to be managed by this controller do not need this step.
- SD does not require any other ServiceControl Manager filesets.

2) (Optional) Enable the GUI interfaces for remote operations by creating the **.sdkey** file on the controller. Use this command:

```
touch /var/adm/sw/.sdkey
```

(This step is not required when you use SD from within the HP ServiceControl Manager.)

NOTE: You can also set up remote access by using the *swacl*(1M) command directly on the remote machines to grant root or non-root access to users from the controller system.

Interactive Operations

The **sd** command is an interactive interface for monitoring and scheduling software jobs. It provides the same functionality as the **swjob** command. You can also use **sd** to invoke the **swinstall**, **copy**, and **swremove** GUIs.

If you have enabled SD's remote operations features, **swinstall**, **swcopy**, and **swremove** provide enhanced GUIs to support operations on remote targets. See **Remote Operation** above for details about enabling remote operations and the enhanced GUIs.

Options

When no options or operands are specified, **swjob** lists the jobs that exist on the local host. These jobs may be pending, active, in the background or completed. The **swjob** command supports the following options:

XToolKit Options
: The **sd** command supports a subset of the standard XToolkit options to control the appearance of the system GUI. The supported options are: **-bg**, **-background**, **-fg**, **-foreground**, **-display**, **-name**, **-xrm**, and **-synchronous**. See the *X*(1) man page by typing **man X** for a definition of these options.

-i Runs the command in interactive mode (invokes the GUI.) (Using this option is an alias for the **sd** command.) See the **Interactive Operation** and **Remote Operation** headings above for details.

-R Applies to target lists as a shorthand for **@ *::***.

-u Causes **swjob** to remove the specified job(s).

-v Causes **swjob** to list all available attributes, one per line. The option applies only to the default list.

-a *attribute* Each job has its own set of attributes. These attributes include such things as job title, schedule date, or results. The **-a** option selects a specific attribute to display. You can specify multiple **-a** options to display multiple attributes. See also *sd*(4) for details on these attributes. This option applies only to the default list.

The logfiles summarizing a job or detailing target actions can be displayed using **-a log**, if **-a log** is specified and no other attribute is specified (i.e. no other attribute may be specified).

-C *session_file*
: Save the current options and operands to *session_file*. You can enter a relative or absolute path with the file name. The default directory for session files is **$HOME/.sw/sessions/**. You can recall a session file with the **-S** option.

-f *jobid_file* Read the list of *jobids* from *jobid_file* instead of (or in addition to) the command line.

-t *target_file* Read the list of *target_selections* from *target_file* instead of (or in addition to) the command line.

-x *option=value*
: Set the session *option* to *value* and override the default value (or a value in an alternate *option_file* specified with the **-X** option). Multiple **-x** options can be specified.

-S *session_file*
: Execute **swjob** based on the options and operands saved from a previous session, as defined in *session_file*. You can save session information to a file with the **-C** option.

-X *option_file* Read the session options and behaviors from *option_file*.

Operands

The **swjob** command supports two types of operands: *jobid* followed by *target selections*. These operands are separated by the "@" (at) character. This syntax implies that the command operates on "jobid at targets".

- jobid The **swjob** command supports the following syntax for each job id:

  jobid

- target selections The **swjob** command supports the following syntax for each target selection:

  [*host*][**:**][*directory*]

EXTERNAL INFLUENCES

Default Options

In addition to the standard options, several SD behaviors and policy options can be changed by editing the default values found in:

/var/adm/sw/defaults the system-wide default values.

$HOME/.swdefaults the user-specific default values.

Values must be specified in the defaults file using this syntax:

[*command_name*.]*option*=*value*

The optional *command_name* prefix denotes one of the SD commands. Using the prefix limits the change in the default value to that command. If you leave the prefix off, the change applies to all commands.

You can also override default values from the command line with the **-x** or **-X** options:

command **-x** *option*=*value*

command **-X** *option_file*

The following section lists all of the keywords supported by the **swjob** command. If a default value exists, it is listed after the "=".

The policy options that apply to **swjob** are:

admin_directory=/var/adm/sw (for normal mode)
admin_directory=/var/home/LOGNAME/sw (for nonprivileged mode)

The location for SD logfiles and the default parent directory for the installed software catalog. The default value is **/var/adm/sw** for normal SD operations. When SD operates in nonprivileged mode (that is, when the **run_as_superuser** default option is set to **true**):

- The default value is forced to **/var/home/LOGNAME/sw**.
- The path element **LOGNAME** is replaced with the name of the invoking user, which SD reads from the system password file.
- If you set the value of this option to **HOME/***path*, SD replaces **HOME** with the invoking user's home directory (from the system password file) and resolves *path* relative to that directory. For example, **HOME/my_admin** resolves to the **my_admin** directory in your home directory.

SD's nonprivileged mode is intended only for managing applications that are specially designed and packaged. This mode cannot be used to manage the HP-UX operating system or patches to it. For a full explanation of nonprivileged SD, see the *Software Distributor Administration Guide*, available at the **http://docs.hp.com** web site.

See also the **run_as_superuser** option.

agent_timeout_minutes=10000

Causes a target agent to exit if it has been inactive for the specified time. This can be used to make target agents more quickly detect lost network connections since RPC can take as long as 130 minutes to detect a lost connection. The recommended value is the longest period of inactivity expected in your environment. For command line invocation, a value between 10 minutes and 60 minutes is suitable. A value of 60 minutes or more is recommended when the GUI will be used. The default of 10000 is slightly less than 7 days.

one_liner={jobid operation state progress results title}

Defines the attributes which will be listed for each job when no **-a** option is specified. Each attribute included in the **one_liner** definition is separated by <tab> or <space>. Any attributes, except **log** may be included in the **one_liner** definition. If a particular attribute does not exist for an object, that attribute is silently ignored.

rpc_binding_info=ncacn_ip_tcp:[2121] ncadg_ip_udp:[2121]

Defines the protocol sequence(s) and endpoint(s) on which the daemon listens and the other commands contact the daemon. If the connection fails for one protocol sequence, the next is attempted. SD supports both the tcp (ncacn_ip_tcp:[2121]) and udp (ncadg_ip_udp:[2121]) protocol sequence on most platforms. See the *sd(5)* man page by typing **man 5 sd** for more information.

rpc_timeout=5

Relative length of the communications timeout. This is a value in the range from 0 to 9 and is interpreted by the DCE RPC. Higher values mean longer times; you may need a higher value for a slow or busy network. Lower values will give faster recognition on attempts to contact hosts that are not up or not running **swagentd**. Each value is approximately twice as long as the preceding value. A value of 5 is about 30 seconds for the

ncadg_ip_udp protocol sequence. This option may not have any noticeable impact when using the **ncacn_ip_tcp** protocol sequence.

run_as_superuser=true

This option controls SD's nonprivileged mode. This option is ignored (treated as true) when the invoking user is super-user.

When set to the default value of true, SD operations are performed normally, with permissions for operations either granted to a local super-user or set by SD ACLs. (See *swacl*(1M) for details on ACLs.)

When set to false and the invoking user is local and is *not* super-user, nonprivileged mode is invoked:

- Permissions for operations are based on the user's file system permissions.
- SD ACLs are ignored.
- Files created by SD have the uid and gid of the invoking user, and the mode of created files is set according to the invoking user's umask.

SD's nonprivileged mode is intended only for managing applications that are specially designed and packaged. This mode cannot be used to manage the HP-UX operating system or patches to it. For a full explanation of nonprivileged SD, see the *Software Distributor Administration Guide*, available at the **http://docs.hp.com** web site.

See also the **admin_directory** option.

targets=

Defines the default *target_selections*. There is no supplied default. If there is more than one target selection, they must be separated by spaces.

verbose=0

Controls the verbosity of the output (stdout). A value of

0 disables output to stdout. (Error and warning messages are always written to stderr).
1 enables verbose messaging to stdout.

Session File

Each invocation of the **swjob** command defines a job display session. The invocation options, source information, software selections, and target hosts are saved before the installation or copy task actually commences. This lets you re-execute the command even if the session ends before proper completion.

Each session is automatically saved to the file **$HOME/.sw/sessions/swjob.last.** This file is overwritten by each invocation of **swjob**.

You can also save session information to a specific file by executing **swjob** with the **-C** *session__file* option.

A session file uses the same syntax as the defaults files. You can specify an absolute path for the session file. If you do not specify a directory, the default location for a session file is **HOME/.sw/sessions/**.

To re-execute a session file, specify the session file as the argument for the **-S** *session__file* option of **swjob**.

Note that when you re-execute a session file, the values in the session file take precedence over values in the system defaults file. Likewise, any command line options or parameters that you specify when you invoke **swjob** take precedence over the values in the session file.

Environment Variables

SD programs are affected by external environment variables.

SD programs that execute control scripts set environment variables for use by the control scripts. **swjob** does not set environmental variables, but it uses them.

Environment variables that affect the SD commands:

LANG Determines the language in which messages are displayed. If LANG is not specified or is set to the empty string, a default value of **C** is used. See the *lang*(5) man page by typing **man 5 lang** for more information.

NOTE: The language in which the SD agent and daemon log messages are displayed is set by the system configuration variable script, **/etc/rc.config.d/LANG**. For example,

/etc/rc.config.d/LANG, must be set to **LANG=ja_JP.SJIS** or **LANG=ja_JP.eucJP** to make the agent and daemon log messages display in Japanese.

LC_ALL Determines the locale to be used to override any values for locale categories specified by the settings of **LANG** or any environment variables beginning with **LC_**.

LC_CTYPE
Determines the interpretation of sequences of bytes of text data as characters (e.g., single- versus multibyte characters in values for vendor-defined attributes).

LC_MESSAGES
Determines the language in which messages should be written.

LC_TIME
Determines the format of dates (*create_date* and *mod_date*) when displayed by **swlist**. Used by all utilities when displaying dates and times in **stdout**, **stderr**, and **logging**.

TZ Determines the time zone for use when displaying dates and times.

Signals

The **swjob** command catches the signals SIGQUIT and SIGINT. If these signals are received, **swjob** prints a message, sends a Remote Procedure Call (RPC) to the daemons to wrap up, and then exits.

Each agent will complete the list task before it wraps up.

OPERATION

Different views of the job information are available. The types of listings that can be selected are given below.

- Default Listing
- Target Listing
- Logfile Listing

Default Listing

If **swjob** is invoked with no options or operands, it lists all jobs that are on the local host. This listing contains one line for each job. The line includes the job tag attribute and all other attributes selected via the **one_liner** option.

Listing jobs on a remote controller is not supported. If a *jobid* is given, information for only that job is displayed.

Status Listing

If a *-R* or **@** *target_specification* is given, the targets for that job and their status are displayed. By default the status information includes Type, State, Progress and Results.

Logfile Listing

One of the attributes "log" encompasses a variety of logfile types. The type of logfile returned when the **-a log** attribute is given depends on the operands given. The types of logfiles:

No target_selections	Show the controller logfile (default).
@ target	Show the agent logfile.

S

RETURN VALUES

The **swjob** command returns:

0 The job information was successfully listed or the job was successfully removed.
1 The list /remove operation failed for all *jobids*.
2 The list /remove operation failed for some *jobids*.

DIAGNOSTICS

The **swjob** command writes to stdout, stderr, and to the agent logfile.

Standard Output

All listings are printed to stdout.

Standard Error

The **swjob** command writes messages for all WARNING and ERROR conditions to stderr.

Logging

The **swjob** command does not log summary events. It logs events about each read task to the **swagent** logfile associated with each *target_selection*.

EXAMPLES

To list all of the jobs that exist on the local host:

```
swjob
```

To show the scheduled date for job hostA-0001:

```
swjob -a schedule hostA-0001
```

For job hostA-0001 list the targets and their status:

```
swjob -R hostA-0001
```

or

```
swjob hostA-0001 @ *
```

For job hostA-0001 list the controller log:

```
swjob -a log hostA-0001
```

For job hostA-0001 list the targetA agent log:

```
swjob -a log targetA-0001 @ targetA
```

FILES

$HOME/.swdefaults
Contains the user-specific default values for some or all SD options.

/usr/lib/sw/sys.defaults
Contains the master list of current SD options (with their default values).

/var/adm/sw/
The directory which contains all of the configurable (and non-configurable) data for SD. This directory is also the default location of logfiles.

/var/adm/sw/defaults
Contains the active system-wide default values for some or all SD options.

/var/adm/sw/queue/
The directory which contains the information about all active and complete install jobs, copy jobs, and other jobs initiated by the SD commands.

AUTHOR

swjob was developed by the Hewlett-Packard Company.

SEE ALSO

swacl(1M), swagentd(1M), swask(1M), swconfig(1M), swcopy(1M), swinstall(1M), swlist(1M), swmodify(1M), swpackage(1M), swreg(1M), swremove(1M), swverify(1M), install-sd(1M), sd(4), swpackage(4), sd(5).

Software Distributor Administration Guide, available at **http://docs.hp.com**.

SD customer web site at **http://software.hp.com/SD_AT_HP/**.

NAME

swlist - display information about software products

SYNOPSIS

swlist [**-d**|**-r**] [**-i**] [**-R**] [**-v**] [**-a** *attribute*] [**-C** *session_file*] [**-f** *software_file*] [**-l** *level*] [**-s** *source*] [**-S** *session_file*] [**-t** *target_file*] [**-x** *option=value*] [**-X** *option_file*] [*software_selections*] [**@** *target_selections*]

Remarks

- This command supports operation on remote systems. See **Remote Operation** below.
- **swlist** supports an interactive user interface that can be invoked by the **swlist -i** command. See **Interactive Operation** below.
- For an overview of all SD commands, see the *sd*(5) man page by typing **man 5 sd** on the command line.

DESCRIPTION

The **swlist** command displays information about software products installed at or available from the specified *target_selections*. It supports these features:

- Specify bundles, products, subproducts, and/or filesets to list.
- Display the files contained in each fileset.
- Display a table of contents from a software source.
- Specify the attributes to display for each software object.
- Display all attributes for bundles, products, subproducts, filesets and/or files.
- Display the full **software_spec** to be used with software selections.
- Display the **readme** file for products.
- Display the depots on a specified host.
- Create a list of products, subproducts, and/or filesets to use as input to the other commands.
- List the categories of available or applied patches.
- List applied patches and their state (applied or committed).

Remote Operation

You can enable Software Distributor (SD) to manage software on remote systems. To let the root user from a central SD *controller* (also called the *central management server* or *manager node*) perform operations on a remote *target* (also called the *host* or *agent*):

1) Install a special HP ServiceControl Manager fileset on the remote systems. This enables remote operations by automatically setting up the root, host, and template Access Control Lists (ACLs) on the remote machines and permitting root access from the controller system. To install the fileset, run the following command on each remote system:

swinstall -s *controller***:/var/opt/mx/depot11 AgentConfig.SD-CONFIG**

NOTES:

- *controller* is the name of the central management server.
- If the target is running HP-UX 10.20, use the same command but substitute **depot10** for **depot11**.
- Targets previously set up by SD/OV to be managed by this controller do not need this step.
- SD does not require any other ServiceControl Manager filesets.

2) (Optional) Enable the GUI interfaces for remote operations by creating the **.sdkey** file on the controller. Use this command:

```
touch /var/adm/sw/.sdkey
```

(This step is not required when you use SD from within the HP ServiceControl Manager.)

S

NOTE: You can also set up remote access by using the *swacl*(1M) command directly on the remote machines to grant root or non-root access to users from the controller system.

Interactive Operation

swlist supports an optional graphical user interface (GUI). (If your terminal or display cannot support the GUI, the command also provides a terminal user interface, in which screen navigation is done with the keyboard and no mouse.)

To invoke the GUI, type:

> **swlist -i** or add the **-i** option with any other command-line options when you invoke the **swlist**.

Previewing Product and OS Update Information

To preview information about new software in the depot, you can use **swlist** to view the **readme** file for each product, including OS update information contained in the SD (SW-DIST product) **readme**. For example, to display the latest OS update information:

```
swlist -d -a readme -l product SW-DIST @ hostA:/depot11
```

Options

When no options or operands are specified, **swlist** lists the software bundles (and products which are not part of a bundle) that are installed at the local host. **swlist** supports the following options:

-d List software available from a depot (instead of software installed on a root filesystem).

-i Invoke the **swlist** interactive user interface. The interactive interface lets you browse SD software objects. Invoking **swlist -i -d** lets you browse depot software. See the **Interactive Operation** and **Remote Operation** headings above for additional details.

-r Operates on an alternate root directory, which must be specified the @ *target_selections* option. (This option is not required for alternate root operations but is maintained for backward compatibility. See the **Alternate Root Directory and Depot Directory** heading in *sd*(5) for more information.)

-R Shorthand for *-l bundle -l product -l subproduct -l fileset*.

-v List all the attributes for an object if no **-a** options are specified. (Vendor-defined attributes are not included. See the **-a** option.) The output lists one attribute per line in the format:

```
attribute_name  attribute_value
```

(See *sd*(4) for details on all SD attributes.)

-a *attribute*
Display a specific *attribute*, such as revision, description, vendor information, size, vendor-defined attributes, or others. (See *sd*(4) for details on all SD attributes.) The output lists one attribute per line in the format:

```
attribute_name  attribute_value
```

To display multiple attributes, specify multiple **-a** options.

To list the full set of attributes for a software object, use the **-v** option.

Note that the **tag** attribute (i.e. the identifier) is always displayed for product, subproduct, and fileset objects. The **path** attribute (i.e. the filename) is always displayed for file objects.

-c *catalog*
Write full catalog structure information into the directory specified by the *catalog* modifier. You can use this exported catalog structure for distributions and to list installed software catalog information.

If you use the **-c** *catalog* option, the **-a** *attribute* and **-l** *level* do not apply. All attributes down to the file level and the control scripts are written to the catalog.

-C *session_file*
Save the current options and operands to *session_file*. You can enter a relative or absolute

path with the file name. The default directory for session files is **/.sw/sessions/**. You can recall a session file with the **-S** option. (Note that session management does not apply to the **swlist** interactive user interface invoked by the **-i** option.)

-f *software_file*
Read the list of *software_selections* from *software_file* instead of (or in addition to) the command line.

-l *level* List all objects down to the specified *level*. Both the specified level(s) and the depth of the specified *software_selections* control the depth of the **swlist** output.

-s *source*
Specify the software source to list. This is an alternate way to list a source depot. Sources can also be specified as target depots and listed using the **-d** option.

-S *session_file*
Execute **swlist** based on the options and operands saved from a previous session, as defined in *session_file*. You can save session information to a file with the **-C** option. (Note that session management does not apply to the **swlist** interactive user interface invoked by the **-i** option.)

-t *target_file*
Read the list of *target_selections* from *target_file* instead of (or in addition to) the command line.

-x *option=value*
Set the session *option* to *value* and override the default value (or a value in an alternate *option_file* specified with the **-X** option). Multiple **-x** options can be specified.

-X *option_file*
Read the session options and behaviors from *option_file*.

Operands

swlist supports two types of operands: *software selections* followed by *target selections*. These operands are separated by the "@" (at) character. This syntax implies that the command operates on "software selections at targets".

Software Selections

swlist supports the following syntax for each *software_selection*:

bundle[**.***product*[**.***subproduct*][**.***fileset*]][**,***version*]

product[**.***subproduct*][**.***fileset*][**,***version*]

- The **=** (equals) relational operator lets you specify selections with the following shell wildcard and pattern-matching notations:

  [ ], *****, **?**

- *Bundles* and *subproducts* are recursive. *Bundles* can contain other *bundles* and *subproducts* can contain other *subproducts*.
- The **\*** software specification selects all products. Use this specification with caution.

The **version** component has the form:

[**,r** <*op*> *revision*][**,a** <*op*> *arch*][**,v** <*op*> *vendor*]
[**,c** <*op*> *category*][**,l=***location*][**,fr** <*op*> *revision*]
[**,fa** <*op*> *arch*]

- *location* applies only to installed software and refers to software installed to a location other than the default product directory.
- **fr** and **fa** apply only to filesets.
- The <*op*> (relational operator) component can be of the form:

  =, **==**, **>=**, **<=**, **<**, **>**, or **!=**

  which performs individual comparisons on dot-separated fields.

  For example, **r>=B.10.00** chooses all revisions greater than or equal to **B.10.00**. The system compares each dot-separated field to find matches.

S

- The **=** (equals) relational operator lets you specify selections with the shell wildcard and pattern-matching notations:

  [ ], *****, **?**, **!**

  For example, the expression **r=1[01].*** returns any revision in version 10 or version 11.
- All version components are repeatable within a single specification (e.g. **r>=A.12**, **r<A.20**). If multiple components are used, the selection must match all components.
- Fully qualified software specs include the **r=**, **a=**, and **v=** version components even if they contain empty strings. For installed software, **l=** is also included.
- No space or tab characters are allowed in a software selection.
- The software *instance_id* can take the place of the version component. It has the form:

  [*instance_id*]

  within the context of an exported catalog, where *instance_id* is an integer that distinguishes versions of products and bundles with the same tag.

Target Selections

swlist supports this syntax for each *target_selection*.

[*host*][**:**][**/**_directory_]

The **:** (colon) is required if both a host and directory are specified.

EXTERNAL INFLUENCES

Default Options

In addition to the standard options, several SD behaviors and policy options can be changed by editing the default values found in:

/var/adm/sw/defaults	the system-wide default values.
$HOME/.swdefaults	the user-specific default values.

Values must be specified in the defaults file using this syntax:

[*command_name*.]*option*=*value*

The optional *command_name* prefix denotes one of the SD commands. Using the prefix limits the change in the default value to that command. If you leave the prefix off, the change applies to all commands.

You can also override default values from the command line with the **-x** or **-X** options:

command **-x** *option*=*value*

command **-X** *option_file*

The following section lists all of the keywords supported by the **swlist** commands. If a default value exists, it is listed after the "=".

The policy options that apply to **swlist** are:

admin_directory=/var/adm/sw (for normal mode)
admin_directory=/var/home/LOGNAME/sw (for nonprivileged mode)

The location for SD logfiles and the default parent directory for the installed software catalog. The default value is **/var/adm/sw** for normal SD operations. When SD operates in nonprivileged mode (that is, when the **run_as_superuser** default option is set to **true**):

- The default value is forced to **/var/home/LOGNAME/sw**.
- The path element **LOGNAME** is replaced with the name of the invoking user, which SD reads from the system password file.
- If you set the value of this option to **HOME/**_path_, SD replaces **HOME** with the invoking user's home directory (from the system password file) and resolves *path* relative to that directory. For example, **HOME/my_admin** resolves to the **my_admin** directory in your home directory.
- If you set the value of the **installed_software_catalog** default option to a relative path, that path is resolved relative to the value of this option.

S

SD's nonprivileged mode is intended only for managing applications that are specially designed and packaged. This mode cannot be used to manage the HP-UX operating system or patches to it. For a full explanation of nonprivileged SD, see the *Software Distributor Administration Guide*, available at the **http://docs.hp.com** web site.

See also the **installed_software_catalog** and **run_as_superuser** options.

agent_timeout_minutes=10000
Causes a target agent to exit if it has been inactive for the specified time. This can be used to make target agents more quickly detect lost network connections since RPC can take as long as 130 minutes to detect a lost connection. The recommended value is the longest period of inactivity expected in your environment. For command line invocation, a value between 10 minutes and 60 minutes is suitable. A value of 60 minutes or more is recommended when the GUI will be used. The default of 10000 is slightly less than 7 days.

codeword=
Provides the "codeword" needed to unlock protected HP CD-ROM software.

Some HP software products are shipped on CD-ROM as "protected" products. That is, they cannot be installed or copied unless a "codeword" and "customer ID" are provided. The codeword is found on the CD-ROM certificate which you received from HP. You may use this default specification on the command line or the SD-UX interactive user interface to enter the codeword.

This default stores the codeword for future reference; it needs to be entered only once. If a new HP product is purchased and a previous codeword has already been entered for that CD-ROM, just enter the new codeword as usual and the codewords will be merged internally.

NOTE: For HP-UX B.10.10 and later systems, SD searches the **.codewords** file on the server that is providing protected software to other hosts. It looks for valid customer_id/codeword pairs. In doing so, SD eliminates the need to enter codewords and customer_ids on every host that is "pulling" the software.

To properly store the customer_id/codeword for a CD-ROM, run **swinstall -p** or **swcopy -p** on the host serving the CD-ROM. After the codeword has been stored, clients installing or copying software using that host and CD-ROM as a source will no longer require a codeword or customer_id.

create_time_filter=0
For cumulative source depots, this option allows consistent software selections over time by **swlist**, **swcopy**, and **swinstall**. The default of zero includes all bundles, products, subproducts, and filesets in the source depot as candidates for selection (and autoselection of dependencies and patches), based on the software selections and other options. When set to a time (specified as seconds from epoch), only those bundles, products, and filesets (and the subproducts in the product) with a create_time less than or equal to the specified value are available for selection (or autoselection). To list the create_time of bundles, products and filesets, use:

```
swlist -a create_time -a create_date
```

customer_id=
This number, also printed on the Software Certificate, is used to "unlock" protected software and restrict its installation to a specific site or owner. It is entered using the **-x** *customer_id=* option or by using the interactive user interface. The *customer_id* can be used on any HP-UX 10.X or later system.

distribution_target_directory=/var/spool/sw
Defines the default location of the target depot.

installed_software_catalog=products
Defines the directory path where the Installed Products Database (IPD) is stored. This information describes installed software. When set to an absolute path, this option defines the location of the IPD. When this option contains a relative path, the SD controller appends the value to the value specified by the **admin_directory** option to determine the path to the IPD. For alternate roots, this path is resolved relative to the location of the alternate root. This option does not affect where software is installed, only the IPD location.

This option permits the simultaneous installation and removal of multiple software applications by multiple users or multiple processes, with each application or group of applications using a different IPD.

Caution: use a specific **installed_software_catalog** to manage a specific application. SD does not support multiple descriptions of the same application in multiple IPDs.

See also the **admin_directory** and **run_as_superuser** options, which control SD's nonprivileged mode. (This mode is intended only for managing applications that are specially designed and packaged. This mode cannot be used to manage the HP-UX operating system or patches to it. For a full explanation of nonprivileged SD, see the *Software Distributor Administration Guide*, available at the **http://docs.hp.com** web site.)

layout_version=1.0

Specifies the POSIX **layout_version** to which the SD commands conform when writing distributions and **swlist** output. Supported values are "1.0" (default) and "0.8".

SD object and attribute syntax conforms to the *layout_version 1.0* specification of the *IEEE POSIX 1387.2 Software Administration* standard. SD commands still accept the keyword names associated with the older layout version, but you should use **layout_version=0.8** only to create distributions readable by older versions of SD.

See the description of the **layout_version** option in *sd*(5) for more information.

level= Specify the *level* of the object to list.

The supported software levels are:

bundle	Show all objects down to the bundle level.
product	Show all objects down to the product level. Also use *-l bundle -l product* to show bundles.
subproduct	Show all objects down to the subproduct level.
fileset	Show all objects down to the fileset level. Also use *-l fileset -l subproduct* to show subproducts.
file	Show all objects down to the file level.
control_file	Show all objects down to the control_file level.
category	Show all categories of available patches.
patch	Show all applied patches.

The supported depot and root levels are:

depot	Show only the depot level (i.e. depots which exist at the specified target hosts).
root	List all alternate roots.
shroot	List all registered shared roots (HP-UX 10.X only).
prroot	List all registered private roots (HP-UX 10.X only).

one_liner=revision title

Defines the attributes which will be listed for each object when no **-a** or **-v** options are specified. Each attribute included in the **one_liner** definition is separated by <tab> or <space>. Any attributes may be included in the **one_liner** definition. If a particular attribute does not exist for an object, that attribute is silently ignored. For example, the **description** attribute is valid for products, subproducts, and filesets, but the **architecture** attribute is only valid for products.

patch_one_liner=revision title patch_state

Specifies the attributes displayed for each object listed when the **-l patch** option is invoked and when no **-a** or **-v** option is specified. The default display attributes are **title** and **patch_state**.

rpc_binding_info=ncacn_ip_tcp:[2121] ncadg_ip_udp:[2121]

Defines the protocol sequence(s) and endpoint(s) on which the daemon listens and the other commands contact the daemon. If the connection fails for one protocol sequence, the next is attempted. SD supports both the tcp (ncacn_ip_tcp:[2121]) and udp (ncadg_ip_udp:[2121]) protocol sequence on most platforms. See the **sd.5** man page by typing **man 5 sd** for more information.

rpc_timeout=5

Relative length of the communications timeout. This is a value in the range from 0 to 9 and

S

is interpreted by the DCE RPC. Higher values mean longer times; you may need a higher value for a slow or busy network. Lower values will give faster recognition on attempts to contact hosts that are not up or not running **swagentd**. Each value is approximately twice as long as the preceding value. A value of 5 is about 30 seconds for the **ncadg_ip_udp** protocol sequence. This option may not have any noticeable impact when using the **ncacn_ip_tcp** protocol sequence.

run_as_superuser=true
This option controls SD's nonprivileged mode. This option is ignored (treated as true) when the invoking user is super-user.

When set to the default value of true, SD operations are performed normally, with permissions for operations either granted to a local super-user or set by SD ACLs. (See *swacl*(1M) for details on ACLs.)

When set to false and the invoking user is local and is *not* super-user, nonprivileged mode is invoked:

- Permissions for operations are based on the user's file system permissions.
- SD ACLs are ignored.
- Files created by SD have the uid and gid of the invoking user, and the mode of created files is set according to the invoking user's umask.

SD's nonprivileged mode is intended only for managing applications that are specially designed and packaged. This mode cannot be used to manage the HP-UX operating system or patches to it. For a full explanation of nonprivileged SD, see the *Software Distributor Administration Guide*, available at the **http://docs.hp.com** web site.

See also the **admin_directory** and **installed_software_catalog** options.

show_superseded_patches=false
Displays or hides superseded patches in **swlist** output. In the default state of **false**, **swlist** will not display superseded patches even if you perform a **swlist** command on the superseded patch. Setting this option to **true** permits display of superseded patches.

select_local=true
If no *target_selections* are specified, select the default **target_directory** of the local host as the *target_selection* for the command.

software=
Defines the default *software_selections*. There is no supplied default. If there is more than one software selection, they must be separated by spaces.

software_view=all_bundles
Indicates the software view to be used as the default level for the software listing in the GUI. It can be set to **all_bundles**, **products**, or a bundle category tag (to indicate to show only bundles of that category).

S

targets=
Defines the default *target_selections*. There is no supplied default (see **select_local** above). If there is more than one target selection, they must be separated by spaces.

verbose=0
Controls how attribute values are displayed. A value of

0 displays only the attribute value.
1 displays both the attribute keyword and value. (See the **-v** option above.)

Session File

Each invocation of **swlist** defines a task session. The command automatically saves options, source information, software selections, and target selections before the task actually commences. This lets you re-execute the command even if the session ends before the task is complete. You can also save session information from interactive or command-line sessions.

Session information is saved to the file **$HOME/.sw/sessions/swlist.last.** This file is overwritten by each invocation of the command. The file uses the same syntax as the defaults files.

From an interactive session, you can save session information into a file at any time by selecting the *Save Session* or *Save Session As* option from the *File* menu.

From a command-line session, you can save session information by executing the command with the **-C***session__file* option. You can specify an absolute path for a session file. If you do not specify a directory, the default location is **$HOME/.sw/sessions/**.

To re-execute a saved session from an interactive session, use the *Recall Session* option from the *File* menu.

To re-execute a session from a command-line, specify the session file as the argument for the **-S** option.

When you re-execute a session file, the values in the session file take precedence over values in the system defaults file. Likewise, any command-line options and parameters take precedence over the values in the session file.

Environment Variables

The environment variable that affects the **swlist** command is:

LANG Determines the language in which messages are displayed. If LANG is not specified or is set to the empty string, a default value of **C** is used. See *lang*(5) for more information.

NOTE: The language in which the SD agent and daemon log messages are displayed is set by the system configuration variable script, **/etc/rc.config.d/LANG**. For example, **/etc/rc.config.d/LANG**, must be set to **LANG=ja_JP.SJIS** or **LANG=ja_JP.eucJP** to make the agent and daemon log messages display in Japanese.

LC_ALL Determines the locale to be used to override any values for locale categories specified by the settings of **LANG** or any environment variables beginning with **LC_**.

LC_CTYPE
Determines the interpretation of sequences of bytes of text data as characters (e.g., single- versus multibyte characters in values for vendor-defined attributes).

LC_MESSAGES
Determines the language in which messages should be written.

LC_TIME
Determines the format of dates (*create_date* and *mod_date*) when displayed by **swlist**. Used by all utilities when displaying dates and times in **stdout**, **stderr**, and **logging**.

TZ Determines the time zone for use when displaying dates and times.

Signals

The **swlist** command catches the signals SIGQUIT and SIGINT. If these signals are received, **swlist** prints a message, sends a Remote Procedure Call (RPC) to the agents to wrap up, and then exits.

Each agent will complete the list task (if the execution phase has already started) before it wraps up.

OPERATION

The output from **swlist** follows this rule with all options: only the lowest level listed (product, subproduct, fileset or file) will be uncommented. Among other things, this allows the output from **swlist** to be used as input to other commands. The one exception is the list that contains files; file-level output is not accepted by other commands.

The types of listings that can be selected are given below. Some of these listings are not exclusive choices, but rather ways to view the objects while controlling the amount of output.

- Default Listing
- Software Listing
- Root Listing
- Depot Listing
- Multiple Targets Listing
- Verbose Listing

Default Listing

If **swlist** is invoked with no *software_selections* and no *target_selections*, a listing of all installed products on the local host is produced. This listing contains one line for each product. The line includes the product tag attributes and all other attributes selected via the **one_liner** option.

If *target_selections* (i.e. target hosts) are specified, this same format listing is produced for the installed software at each of the specified hosts.

Software Listing

A listing of software objects is controlled by the specified *software_selections*, and also by the **-l** option (**swlist.level=**). **swlist** lists the contents of each software object specified in the *software_selections*. For example, if you specify product selections, the subproducts and/or filesets contained immediately below each product will be listed. If you specify fileset selections, the files contained in each fileset will be listed.

The depth of objects listed is controlled with the **-l** option. This option can expand or restrict the depth in concert with the specified software selections. By default, the contents of a specified software selection are always listed (as described above). The **-l** option can defeat this listing by specifying a level equivalent to the level of objects in the *software_selections*. For example, if you want to list specific product selections but not their contents, use **-l product**. If you want to list specific fileset selections but not their contained files, use **-l fileset**. The *software_selection* options only apply if the level is bundle, product, subproduct, fileset, file, or patch.

Depot Listing

Another class of objects that **swlist** can display are software depots. For example, the user can list all registered depots on a given host. A combination of the **-l depot** option and *target_selections* operands can produce a variety of depot listings.

Multiple Targets Listing

Multiple *target_selections* (i.e. root filesystems, alternate roots, or depots) are listed sequentially: list all the requested objects and attributes from the first *target_selection*, followed by the second *target_selection*, etc.

Verbose Listing

The **-v** option causes a verbose listing to be generated. A verbose listing includes all attributes defined for an object. The **swlist** command prints the keyword and value for each attribute. The attributes are listed one per line. The user can post-process (filter) the output with *grep*(1), *awk*(1), and/or *sed*(1) to get the fields of interest.

The depot's attributes are displayed if **swlist** is called with the **-v** and **-l depot** options, and a specific depot *target_selection*.

Attributes for a particular software level (**product/subproduct/fileset/file**) are displayed based on the depth of the specified *software_selections*. For example, *swlist -v product1.fileset1* will give all fileset attributes for *fileset1*. If the **-v** option is used with the **-l option**, the different listing are:

- To display attributes for all products, use **swlist -v -l product**
- To display attributes for all products and subproducts, use **swlist -v -l subproduct**
- To display attributes for all products and filesets, use **swlist -v -l fileset**
- To display attributes for all products, filesets, and files, use
  swlist -v -l file

RETURN VALUE

The **swlist** command returns:

0	The *software_selections* and/or *target_selections* were successfully listed.
1	The list operation failed on all *target_selections*.
2	The list operation failed on some *target_selections*.

DIAGNOSTICS

The **swlist** command writes to stdout, stderr, and to the agent logfile.

Standard Output

All listings are printed to stdout.

Standard Error

The **swlist** command writes messages for all WARNING and ERROR conditions to stderr.

Logging

The **swlist** command does not log summary events. It logs events about each read task to the **swagent** logfile associated with each *target_selection*.

You can use the **swlist** interactive interface (**swlist -i -d**) to view the **swaudit.log** file.

EXAMPLES

Run the `swlist` interactive interface:

```
swlist -i @ host1
```

Use interactive `swlist` to view a depot:

```
swlist -i -d @ /tmp/depot
```

List all of the products installed on the local host:

```
swlist
```

Generate a comprehensive listing that includes all filesets for the product NETWORKING:

```
swlist -v -l fileset NETWORKING
```

List all the attributes for the ARPA-RUN fileset:

```
swlist -v NETWORKING.ARPA.ARPA-RUN
```

List the C product installed on several remote hosts:

```
swlist cc @ hostA hostB hostC
```

List the FRAME product relocated to directory `/opt` on host1:

```
swlist FRAME,1=/opt @ host1
```

List all the versions of the FRAME product installed on the toolserver host:

```
swlist FRAME @ toolserver
```

List all products in a shared root (HP-UX 10.X only):

```
swlist -r @ /export/shared_roots/OS_700
```

List products in a client's private root (HP-UX 10.X only):

```
swlist -r @  /export/private_roots/client
```

List the contents of the local tape, `/dev/rmt/0m`:

```
swlist -d @ /dev/rmt/0m
```

or, alternatively:

```
swlist -s /dev/rmt/0m
```

List the tag and revision attributes for all products on the local tape `/dev/rmt/0m`:

```
swlist -d -a revision @ /dev/rmt/0m
```

or, alternatively:

```
swlist -a revision -s /dev/rmt/0m @
```

Display the README file for the FRAME product:

```
swlist -a readme FRAME
```

List the products stored in a remote depot:

```
swlist -d @ hostA:/depot
```

List all depots on a host:

```
swlist -l depot @ hostA
```

List the categories defined in the depot mounted at `/CD`.

```
swlist -d -l category @ /CD
```

Output:

```
critical_patch  1.0  Patches to fix system hangs or data corruption
S747_upgrade    2.0  Patches needed to upgrade to an S747
security_patch  2.0  Patches affecting system security
```

List a particular attribute of a category object identified by the tag `critical_patch`.

S

```
swlist -a description -l category critical_patch
```

Use the `swlist -l` option and `patch` level to display the values of a fileset's *applied_patches* attribute.

```
swlist -l patch BogusProduct
```

Output:

```
BogusProduct          1.0             This is a Bogus Product
BogusProduct.FakeFS   Fake fileset
PHZX-0004.FakeFS      Patch for defect X       superseded
PHZX-3452.FakeFS      Patch for defect Y       applied
```

Another example showing just the patch:

```
swlist -l patch PHZX-0004
```

Output:

```
PHZX-0004             1.0             Patch product
PHZX-0004.FakeFS      Patch for defect X       superseded
```

FILES

`$HOME/.swdefaults`
Contains the user-specific default values for some or all SD options.

`$HOME/.sw/sessions/`
Contains session files automatically saved by the SD commands, or explicitly saved by the user.

`/usr/lib/sw/sys.defaults`
Contains the master list of current SD options (with their default values).

`/var/adm/sw/`
The directory which contains all of the configurable (and non-configurable) data for SD. This directory is also the default location of logfiles.

`/var/adm/sw/defaults`
Contains the active system-wide default values for some or all SD options.

`/var/adm/sw/host_object`
The file which stores the list of depots registered at the local host.

`/var/adm/sw/products/`
The Installed Products Database (IPD), a catalog of all products installed on a system.

`/var/spool/sw/`
The default location of a source and target software depot.

AUTHOR

`swlist` was developed by the Hewlett-Packard Company and Mark H. Colburn (see *pax*(1)).

S

SEE ALSO

swacl(1M), swagentd(1M), swask(1M), swconfig(1M), swcopy(1M), swinstall(1M), swjob(1M), swmodify(1M), swpackage(1M), swreg(1M), swremove(1M), swverify(1M), install-sd(1M), sd(4), swpackage(4), sd(5).

Software Distributor Administration Guide, available at **`http://docs.hp.com`**.

SD customer web site at **`http://software.hp.com/SD_AT_HP/`**.

NAME

swmodify - modify software products in a target root or depot

SYNOPSIS

swmodify [**-d**|**-r**] [**-p**] [**-u**] [**-v**] [**-V**] [**-a** *attribute*=[*value*]] [**-c** *catalog*] [**-C** *session_file*] [**-f** *software_file*] [**-P** *pathname_file*] [**-s** *product_specification_file*| [**-S** *session_file*] [**-x** *option=value*] [**-X** *option_file*] [*software_selections*] [**@** *target_selection*]

Remarks

- This command supports operation on remote systems. See **Remote Operation** below.
- For an overview of all SD commands, see the *sd*(5) man page by typing **man 5 sd** on the command line.

DESCRIPTION

The **swmodify** command modifies the definitions of software objects installed into a primary or alternate root, or available from a software depot. It supports the following features:

- adding new objects - The user can add new bundles, products, subproducts, filesets, control files, and files to existing objects (which will contain them).
- deleting existing objects - The user can delete existing bundles, products, subproducts, filesets, control files, and files from the objects which contain them.
- modifying attribute values - The user can add an attribute, delete an attribute, or change the existing value of an attribute for any existing object. When adding a new object, the user can at the same time define attributes for it.
- committing software patches - The user can remove saved backup files, committing the software patch.

With the exception of control files, **swmodify** does not manipulate the actual files that make up a product (fileset). The command manipulates the catalog information which describes the files. However, **swmodify** can replace the contents of control files.

Common uses of **swmodify** include:

- adding file definitions to the existing list of file definitions in a fileset. Example: If a fileset's control scripts add new files to the installed file system, the scripts can call **swmodify** to "make a record" of those new files.
- changing the values of existing attributes. Example: If a product provides a more complex configuration process (beyond the SD configure script), that script can set the fileset's state to CONFIGURED upon successful execution.
- defining new objects. Example: to "import" the definition of an existing application that was not installed by SD, construct a simple PSF describing the product. Then invoke **swmodify** to load the definition of the existing application into the IPD.

S

Remote Operation

You can enable SD to manage software on remote systems. To let the root user from a central SD *controller* (also called the *central management server* or *manager node*) perform operations on a remote *target* (also called the *host* or *agent*):

1) Install a special HP ServiceControl Manager fileset on the remote systems. This enables remote operations by automatically setting up the root, host, and template Access Control Lists (ACLs) on the remote machines and permitting root access from the controller system. To install the fileset, run the following command on each remote system:

swinstall -s *controller***:/var/opt/mx/depot11 AgentConfig.SD-CONFIG**

NOTES:

- *controller* is the name of the central management server.
- If the target is running HP-UX 10.20, use the same command but substitute **depot10** for **depot11**.
- Targets previously set up by SD/OV to be managed by this controller do not need this step.

- SD does not require any other ServiceControl Manager filesets.

2) (Optional) **swinstall**, **swcopy**, and **swremove** have enhanced GUI interfaces for remote operations. Enable the enhanced GUIs by creating the **.sdkey** file on the controller. Use this command:

touch /var/adm/sw/.sdkey

NOTES:

- This step is not required when you use SD from within the HP ServiceControl Manager.
- See *sd(5)*, *swinstall(1M)*, *swcopy(1M)*, *swjob(1M)*, *swlist(1M)*, or *swremove(1M)* for more information on interactive operations.

NOTE: You can also set up remote access by using **swacl** directly on the remote machines to grant root or non-root access to users from the controller system.

Options

swmodify supports the following options:

-d Perform modifications on a depot (not on a primary or alternate root). The given *target_selection* must be a depot.

-p Preview a modify session without modifying anything within the *target_selection*.

-r Performs modifications on an alternate root directory, which must be specified the **@** *target_selections* option. (This option is not required for alternate root operations but is maintained for backward compatibility. See the **Alternate Root Directory and Depot Directory** heading in *sd*(5) for more information.)

-u If no *-a attribute=value* options are specified, then delete the given *software_selections* from within the given *target_selection*. This action deletes the definitions of the software objects from the depot catalog or installed products database.

If *-a attribute* options are specified, then delete these attribute definitions from the given *software_selections* (from within the given *target_selection*).

-v Turn on verbose output to stdout.

-V List the data model revisions that this command supports.

-a *attribute*[**=***value*]

Add, modify, or delete the *value* of the given *attribute*. If the **-u** option is specified, then delete the *attribute* from the given *software_selections* (or delete the *value* from the set of values currently defined for the *attribute*). Otherwise add/modify the *attribute* for each *software_selection* by setting it to the given *value*.

Multiple **-a** options can be specified. Each attribute modification will be applied to every *software_selection*.

The **-s** and **-a** options are mutually exclusive; the **-s** option cannot be specified when the **-a** option is specified.

S

-c *catalog*

Specifies the pathname of the catalog which will be added, modified, or used as input by **swmodify**.

The **-c** and **-a** options are mutually exclusive, the **-c** option cannot be specified when the **-a** option is specified.

-C *session_file*

Save the current options and operands to *session_file*. You can enter a relative or absolute path with the file name. The default directory for session files is **$HOME/.sw/sessions/**. You can recall a session file with the **-S** option.

-f *software_file*

Read the list of *software_selections* from *software_file* instead of (or in addition to) the command line.

-P *pathname_file*

Specify a file containing the pathnames of files being added to or deleted from the IPD instead of having to specify them individually on the command line.

-s *product_specification_file*
The source *Product Specification File* (PSF) describes the product, subproduct, fileset, and/or file definitions which will be added, modified, or used as input by **swmodify**.

The **-s** and **-u** options are mutually exclusive, the **-s** option cannot be specified when the **-u** option is specified.

-S *session_file*
Execute **swmodify** based on the options and operands saved from a previous session, as defined in *session_file*. You can save session information to a file with the **-C** option.

-x *option=value*
Set the session *option* to *value* and override the default value (or a value in an alternate *options_file* specified with the **-X** option). Multiple **-x** options can be specified.

-X *option_file*
Read the session options and behaviors from *options_file*.

Operands

The **swmodify** command supports two types of operands: *software selections* followed by *target selections*. These operands are separated by the "@" (at) character. This syntax implies that the command operates on "software selections at targets".

Software Selections

If a *product_specification_file* is specified, **swmodify** will select the *software_selections* from the full set defined within the PSF. The software selected from a PSF is then applied to the *target_selection*, with the selected software objects either added to it or modified within it. If a PSF is not specified, **swmodify** will select the *software_selections* from the software defined in the given (or default) *target_selection*.

The **swmodify** command supports the following syntax for each *software_selection*:

bundle[**.***product*[**.***subproduct*][**.***fileset*]][**,***version*]

product[**.***subproduct*][**.***fileset*][**,***version*]

- The **=** (equals) relational operator lets you specify selections with the following shell wildcard and pattern-matching notations:

  [ ], *****, **?**

- *Bundles* and *subproducts* are recursive. *Bundles* can contain other *bundles* and *subproducts* can contain other *subproducts*.
- The **\*** software specification selects all products. Use this specification with caution.

The **version** component has the form:

[**,r** <*op*> *revision*][**,a** <*op*> *arch*][**,v** <*op*> *vendor*]
[**,c** <*op*> *category*][**,q=***qualifier*][**,l=***location*]
[**,fr** <*op*> *revision*][**,fa** <*op*> *arch*]

- *location* applies only to installed software and refers to software installed to a location other than the default product directory.
- **fr** and **fa** apply only to filesets.
- The <*op*> (relational operator) component can be of the form:

  =, **==**, **>=**, **<=**, **<**, **>**, or **!=**

  which performs individual comparisons on dot-separated fields.

  For example, **r>=B.10.00** chooses all revisions greater than or equal to **B.10.00**. The system compares each dot-separated field to find matches.
- The **=** (equals) relational operator lets you specify selections with the shell wildcard and pattern-matching notations:

  [ ], *****, **?**, **!**

  For example, the expression **r=1[01].*** returns any revision in version 10 or version 11.
- All version components are repeatable within a single specification (e.g. **r>=A.12**, **r<A.20**). If multiple components are used, the selection must match all components.

S

- Fully qualified software specs include the **r=**, **a=**, and **v=** version components even if they contain empty strings. For installed software, **l=** is also included.
- No space or tab characters are allowed in a software selection.
- The software *instance_id* can take the place of the version component. It has the form:

  [*instance_id*]

  within the context of an exported catalog, where *instance_id* is an integer that distinguishes versions of products and bundles with the same tag.

Target Selection

The **swmodify** command supports the specification of a single, local *target_selection*, using the syntax:

[**@ /***directory*]

When operating on the primary root, no *target_selection* needs to be specified. (The target / is assumed.) When operating on a software depot, the *target_selection* specifies the path to that depot. If the **-d** option is specified and no *target_selection* is specified, the default **distribution_target_directory** is assumed (see below).

EXTERNAL INFLUENCES

Default Options

In addition to the standard options, several SD behaviors and policy options can be changed by editing the default values found in:

/var/adm/sw/defaults	the system-wide default values.
$HOME/.swdefaults	the user-specific default values.

Values must be specified in the defaults file using this syntax:

[*command_name***.**]*option***=***value*

The optional *command_name* prefix denotes one of the SD commands. Using the prefix limits the change in the default value to that command. If you leave the prefix off, the change applies to all commands.

You can also override default values from the command line with the **-x** or **-X** options:

command **-x** *option***=***value*

command **-X** *option_file*

The following keywords are supported by **swmodify**. If a default value exists, it is listed after the "=". The commands that this option applies to are also specified. The policy options that apply to **swmodify** are:

admin_directory=/var/adm/sw (for normal mode)
admin_directory=/var/home/LOGNAME/sw (for nonprivileged mode)

The location for SD logfiles and the default parent directory for the installed software catalog. The default value is **/var/adm/sw** for normal SD operations. When SD operates in nonprivileged mode (that is, when the **run_as_superuser** default option is set to **true**):

- The default value is forced to **/var/home/LOGNAME/sw**.
- The path element **LOGNAME** is replaced with the name of the invoking user, which SD reads from the system password file.
- If you set the value of this option to **HOME/***path*, SD replaces **HOME** with the invoking user's home directory (from the system password file) and resolves *path* relative to that directory. For example, **HOME/my_admin** resolves to the **my_admin** directory in your home directory.
- If you set the value of the **installed_software_catalog** default option to a relative path, that path is resolved relative to the value of this option.

SD's nonprivileged mode is intended only for managing applications that are specially designed and packaged. This mode cannot be used to manage the HP-UX operating system or patches to it. For a full explanation of nonprivileged SD, see the *Software Distributor Administration Guide*, available at the **http://docs.hp.com** web site.

See also the **installed_software_catalog** and **run_as_superuser** options.

compress_index=false
Determines whether SD commands create compressed INDEX and INFO catalog files when writing to target depots or roots. The default of **false** does not create compressed files. When set to **true**, SD creates compressed and uncompressed INDEX and INFO files. The compressed files are named INDEX.gz and INFO.gz, and reside in the same directories as the uncompressed files.

Compressed files can enhance performance on slower networks, although they may increase disk space usage due to a larger Installed Products Database and depot catalog. SD controllers and target agents for HP-UX 11.01 and higher automatically load the compressed INDEX and INFO files from the source agent when:

- The source agent supports this feature.
- INDEX.gz or INFO.gz exist on the source depot.
- INDEX.gz or INFO.gz are not older than the corresponding uncompressed INDEX or INFO files.

The uncompressed INDEX or INFO file is accessed by the source agent if any problem occurs when accessing, transferring, or uncompressing the INDEX.gz or INFO.gz file.

control_files=
When adding or deleting control file objects, this option lists the tags of those control files. There is no supplied default. If there is more than one tag, they must be separated by white space and surrounded by quotes.

distribution_target_directory=/var/spool/sw
Defines the default distribution directory of the target depot. The *target_selection* operand overrides this default.

files= When adding or deleting file objects, this option lists the pathnames of those file objects. There is no supplied default. If there is more than one pathname, they must be separated by white space.

installed_software_catalog=products
Defines the directory path where the Installed Products Database (IPD) is stored. This information describes installed software. When set to an absolute path, this option defines the location of the IPD. When this option contains a relative path, the SD controller appends the value to the value specified by the **admin_directory** option to determine the path to the IPD. For alternate roots, this path is resolved relative to the location of the alternate root. This option does not affect where software is installed, only the IPD location.

This option permits the simultaneous installation and removal of multiple software applications by multiple users or multiple processes, with each application or group of applications using a different IPD.

Caution: use a specific **installed_software_catalog** to manage a specific application. SD does not support multiple descriptions of the same application in multiple IPDs.

See also the **admin_directory** and **run_as_superuser** options, which control SD's nonprivileged mode. (This mode is intended only for managing applications that are specially designed and packaged. This mode cannot be used to manage the HP-UX operating system or patches to it. For a full explanation of nonprivileged SD, see the *Software Distributor Administration Guide*, available at the **http://docs.hp.com** web site.)

layout_version=1.0
Specifies the POSIX **layout_version** to which the SD commands conform when writing distributions and **swlist** output. Supported values are "1.0" (default) and "0.8".

SD object and attribute syntax conforms to the *layout_version 1.0* specification of the *IEEE POSIX 1387.2 Software Administration* standard. SD commands still accept the keyword names associated with the older layout version, but you should use **layout_version=0.8** only to create distributions readable by older versions of SD.

See the description of the **layout_version** option in *sd*(5) for more information.

`log_msgid=0`

Adds numeric identification numbers at the beginning of SD logfile messages:

`0` (default) No identifiers are attached to messages.
`1` Adds identifiers to ERROR messages only.
`2` Adds identifiers to ERROR and WARNING messages.
`3` Adds identifiers to ERROR, WARNING, and NOTE messages.
`4` Adds identifiers to ERROR, WARNING, NOTE, and certain other informational messages.

`logdetail=false`

The **`logdetail`** option controls the amount of detail written to the log file. When set to **`true`**, this option adds detailed task information (such as options specified, progress statements, and additional summary information) to the log file. This information is in addition to log information controlled by the **`loglevel`** option.

`logfile=/var/adm/sw/sw<modify>.log`

Defines the default log file for **`swmodify`**.

`loglevel=1`

Controls the log level for the events logged to the **`swmodify`** logfile, the target agent logfile, and the source agent logfile. This information is in addition to the detail controlled by the **`logdetail`** option. See **`logdetail`** for more information.
A value of

`0` provides no information to the log files.
`1` enables verbose logging to the log files.
`2` enables very verbose logging to the log files.

`patch_commit=false`

Commits a patch by removing files saved for patch rollback. When set to **`true`**, you cannot roll back (remove) a patch unless you remove the associated base software that the patch modified.

`run_as_superuser=true`

This option controls SD's nonprivileged mode. This option is ignored (treated as true) when the invoking user is super-user.

When set to the default value of true, SD operations are performed normally, with permissions for operations either granted to a local super-user or set by SD ACLs. (See *swacl*(1M) for details on ACLs.)

When set to false and the invoking user is local and is *not* super-user, nonprivileged mode is invoked:

- Permissions for operations are based on the user's file system permissions.
- SD ACLs are ignored.
- Files created by SD have the uid and gid of the invoking user, and the mode of created files is set according to the invoking user's umask.

SD's nonprivileged mode is intended only for managing applications that are specially designed and packaged. This mode cannot be used to manage the HP-UX operating system or patches to it. For a full explanation of nonprivileged SD, see the *Software Distributor Administration Guide*, available at the **`http://docs.hp.com`** web site.

See also the **`admin_directory`** and **`installed_software_catalog`** options.

`software=`

Defines the default *software_selections*. There is no supplied default. If there is more than one software selection, they must be separated by spaces. Software is usually specified in a software input file, as operands on the command line, or in the GUI.

`source_file=`

Defines the default location of the source product specification file (PSF). The **`host:path`** syntax is not allowed, only a valid **`path`** can be specified. The **`-s`** option overrides this value.

`targets=`

Defines the default *target_selections*. There is no supplied default (see **`select_local`** above). If there is more than one target selection, they must be separated by spaces.

Targets are usually specified in a target input file, as operands on the command line, or in the GUI.

verbose=1

Controls the verbosity of a non-interactive command's output:

0 disables output to stdout. (Error and warning messages are always written to stderr).
1 enables verbose messaging to stdout.
2 for **swmodify**, enables very verbose messaging to stdout.

Session File

Each invocation of the **swmodify** command defines a modify session. The invocation options, source information, software selections, and target hosts are saved before the installation or copy task actually commences. This lets you re-execute the command even if the session ends before proper completion.

Each session is automatically saved to the file **$HOME/.sw/sessions/swmodify.last**. This file is overwritten by each invocation of **swmodify**.

You can also save session information to a specific file by executing **swmodify** with the **-C** *session__file* option.

A session file uses the same syntax as the defaults files. You can specify an absolute path for the session file. If you do not specify a directory, the default location for a session file is **$HOME/.sw/sessions/**.

To re-execute a session file, specify the session file as the argument for the **-S** *session__file* option of **swmodify**. See the *swpackage*(4) by typing **man 4 swpackage** for PSF syntax.

Note that when you re-execute a session file, the values in the session file take precedence over values in the system defaults file. Likewise, any command line options or parameters that you specify when you invoke **swmodify** take precedence over the values in the session file.

Environment Variables

The environment variable that affects **swmodify** is:

LANG Determines the language in which messages are displayed. If LANG is not specified or is set to the empty string, a default value of **C** is used. See the *lang*(5) man page by typing **man 5 lang** for more information.

NOTE: The language in which the SD agent and daemon log messages are displayed is set by the system configuration variable script, **/etc/rc.config.d/LANG**. For example, **/etc/rc.config.d/LANG**, must be set to **LANG=ja_JP.SJIS** or **LANG=ja_JP.eucJP** to make the agent and daemon log messages display in Japanese.

LC_ALL Determines the locale to be used to override any values for locale categories specified by the settings of **LANG** or any environment variables beginning with **LC_**.

LC_CTYPE

Determines the interpretation of sequences of bytes of text data as characters (e.g., single- versus multibyte characters in values for vendor-defined attributes).

LC_MESSAGES

Determines the language in which messages should be written.

LC_TIME

Determines the format of dates (*create_date* and *mod_date*) when displayed by **swlist**. Used by all utilities when displaying dates and times in **stdout**, **stderr**, and **logging**.

TZ Determines the time zone for use when displaying dates and times.

S

Signals

The **swmodify** command ignores SIGHUP, SIGTERM, SIGUSR1, and SIGUSR2. The **swmodify** command catches SIGINT and SIGQUIT. If these signals are received, **swmodify** prints a message and then exits. During the actual database modifications, **swmodify** blocks these signals (to prevent any data base corruption). All other signals result in their default action being performed.

RETURN VALUES

The **swmodify** command returns:

0 The add, modify, or delete operation(s) were successfully performed on the given *software_selections*.

1 An error occurred during the session (e.g. bad syntax in the PSF, invalid *software_selection*, etc.) Review stderr or the logfile for details.

DIAGNOSTICS

The **swmodify** command writes to stdout, stderr, and to specific logfiles.

Standard Output

In verbose mode, the **swmodify** command writes messages for significant events. These include:

- a begin and end session message,
- selection, analysis, and execution task messages.

Standard Error

The **swmodify** command also writes messages for all WARNING and ERROR conditions to stderr.

Logfile

The **swmodify** command logs events to the command logfile and to the **swmodify** logfile associated with each *target_selection*.

Command Log

The **swmodify** command logs all messages to the the logfile **/var/adm/sw/swmodify.log**. (The user can specify a different logfile by modifying the **logfile** option.)

Target Log

When modifying installed software, **swmodify** logs messages to the file **var/adm/sw/swagent.log** beneath the root directory (e.g. **/** or an alternate root directory). When modifying available software (within a depot), **swmodify** logs messages to the file **swagent.log** beneath the depot directory (e.g. **/var/spool/sw**).

EXAMPLES

Add additional files to an existing fileset:

```
swmodify -xfiles='/tmp/a /tmp/b /tmp/c'  PRODUCT.FILESET
```

Replace the definitions of existing files in an existing fileset (e.g. to update current values for the files' attributes):

```
chown root /tmp/a /tmp/b
swmodify -x files='/tmp/a /tmp/b'  PRODUCT.FILESET
```

Delete control files from a fileset in an existing depot:

```
swmodify -d -u -x control_files='checkinstall subscript' \
         PRODUCT.FILESET @  /var/spool/sw
```

Create a new fileset definition where the description is contained in the PSF file **new_fileset_definition**:

```
swmodify -s new_fileset_definition
```

Delete an obsolete fileset definition:

```
swmodify -u PRODUCT.FILESET
```

Commit a patch (remove files saved for patch rollback):

```
swmodify -x patch_commit=true PATCH
```

Create some new bundle definitions for products in an existing depot:

```
swmodify -d -s new_bundle_definitions  \*   @  /mfg/master_depot
```

Modify the values of some fileset's attributes:

```
swmodify -a state=installed  PRODUCT.FILESET
```

Modify the attributes of a depot:

```
swmodify -a title='Manufacturing's master depot' \
         -a description=</tmp/mfg.description @  /mfg/master_depot
```

S

WARNINGS

If the *target_selection* is a software depot and you delete file definitions from the given *software_selections*, the files' contents are not deleted from the depot.

FILES

`$HOME/.swdefaults`
Contains the user-specific default values for some or all SD options.

`$HOME/.sw/sessions/`
Contains session files automatically saved by the SD commands, or explicitly saved by the user.

`/usr/lib/sw/sys.defaults`
Contains the master list of current SD options (with their default values).

`/var/adm/sw/`
The directory which contains all of the configurable (and non-configurable) data for SD. This directory is also the default location of logfiles.

`/var/adm/sw/defaults`
Contains the active system-wide default values for some or all SD options.

`/var/adm/sw/products/`
The Installed Products Database (IPD), a catalog of all products installed on a system.

`/var/spool/sw/`
The default location of a target software depot.

AUTHOR

`swmodify` was developed by the Hewlett-Packard Company.

SEE ALSO

swacl(1M), swagentd(1M), swask(1M), swconfig(1M), swcopy(1M), swinstall(1M), swjob(1M), swlist(1M), swpackage(1M), swreg(1M), swremove(1M), swverify(1M), install-sd(1M), sd(4), swpackage(4), sd(5).

Software Distributor Administration Guide, available at **`http://docs.hp.com`**.

SD customer web site at **`http://software.hp.com/SD_AT_HP/`**.

NAME

swpackage - package software products into a target depot or tape

SYNOPSIS

swpackage [**-p**] [**-v**] [**-V**] [**-C** *session_file*] [**-d** *directory* | *device*] [**-f** *software_file*] [**-s** *product_specification_file* | *directory*] [**-S** *session_file*] [**-x** *option=value*] [**-X** *option_file*] [*software_selections*] [**@** *target_selection*]

Remarks

- For a description of the Product Specification File (PSF) used as input to the **swpackage** command, see the *swpackage*(4) man page by typing **man 4 swpackage** on the command line.
- For an overview of all SD commands, see the *sd*(5) man page by typing **man 5 sd** on the command line.
- For descriptions of all SD objects, attributes and data formats, see the *sd*(4) man page by typing **man 4 sd** on the command line.

DESCRIPTION

The **swpackage** command is not distributed; it only operates on the local host. It packages software products into:

- a distribution directory (which can be accessed directly or copied onto a CD-ROM),
- a distribution tape, such as DDS, nine-track or cartridge tapes.

A software *product* is organized into a three-level hierarchy: *products*, *subproducts*, and *filesets*. The actual files that make up a product are packaged into filesets. Subproducts can be used to partition or subset the filesets into logical groupings. (Subproducts are optional.) A product, subproduct, and fileset also have attributes associated with them.

Both directory and tape distributions use the same format. The **swpackage** command:

- Organizes the software to be packaged into products, subproducts, and filesets,
- Provides flexible mechanisms to package source files into filesets,
- Modifies existing products in a distribution directory,
- Copies products in a distribution directory to a distribution tape.

Both the **swpackage** and **swcopy** commands create or modify a target depot. The differences between these commands are:

- The **swcopy** command copies products from an existing depot to another depot. The **swpackage** command creates products based on the user's specification, and packages these products into a depot.
- **swpackage** can be used to re-package *software_selections* from an existing distribution directory to a distribution tape.
- The **swcopy** command can copy from a local or remote source to a set of local or remote targets. The **swpackage** command packages source files from the local filesystem into a product, for insertion into a local distribution directory or tape.
- After creating a target depot, **swcopy** registers that directory with the local **swagentd** so that it can be found by **swlist**, **swinstall**, etc. With **swpackage**, the depot is not registered; the user must explicitly invoke the **swreg** command.

Layout Version

By default, SD object and attribute syntax conforms to the *layout_version 1.0* specification of the *IEEE POSIX 1387.2 Software Administration* standard. SD commands still accept the keyword names associated with the older *layout version 0.8*, but you should use the older version only to create distributions readable by older versions of SD.

Which **layout_version** the SD commands write is controlled by the **layout_version** option or by specifying the **layout_version** attribute in the **PSF** file.

See *sd*(4), the description of the **layout_version** option in the following section and in *sd*(5) for more information. See *sd*(4) for more information on **PSF** files.

Options

swpackage supports the following options:

-p Previews a package session without actually creating or modifying the distribution tape.

-v Turns on verbose output to stdout. Verbose output is enabled by default, see the **verbose** option below.

-V List the data model revision that **swpackage** supports. By default, **swpackage** always packages using the latest data model revision.

-C *session_file*
Save the current options and operands to *session_file*. You can enter a relative or absolute path with the file name. The default directory for session files is **$HOME/.sw/sessions/**. You can recall a session file with the **-S** option.

-d *directory* | *device*
(Obsolete but allowed for backward compatibility. Use the **@** *target_selection* operand instead.)

If creating a distribution directory, this option defines the pathname of the *directory*. If creating a distribution tape, this option defines the *device* file on which to write the distribution. When creating a distribution tape, the tape device (file) must exist, and the **-x media_type=tape** option must be specified (see below).

You can also specify that the **swpackage** output be "piped" to an external command using:

```
swpackage -d "| <command>" -x media_type=tape -s <source>
```

The **|** symbol and command must be quoted because it is interpreted by **swpackage** and not the shell.

-f *software_file*
Read the list of *software_selections* from *software_file* instead of (or in addition to) the command line.

-s *product_specification_file* | *directory*
The source PSF describes the product, subproduct, fileset, and file definitions used to build a software product from a set of source files.

The source can also be an existing *directory* depot (which already contains products).

-S *session_file*
Execute **swpackage** based on the options and operands saved from a previous session, as defined in *session_file*. You can save session information to a file with the **-C** option.

-x *option=value*
Set the session *option* to *value* and override the default value (or a value in an alternate *options_file* specified with the **-X** option). Multiple **-x** options can be specified.

-X *option_file*
Read the session options and behaviors from *options_file*.

S

Software Selections

If specified, the software selections cause **swpackage** to only (re)package those software selections from the full set defined in the source *product_specification_file*. If no *software_selections* are specified, then **swpackage** will (re)package all the products defined in the source *product_specification_file*.

The **swpackage** command supports the following syntax for each *software_selection*:

bundle[**.***product*[**.***subproduct*][**.***fileset*]][**,***version*]

product[**.***subproduct*][**.***fileset*][**,***version*]

- The **=** (equals) relational operator lets you specify selections with the following shell wildcard and pattern-matching notations:

  [ ], *****, **?**

- *Bundles* and *subproducts* are recursive. *Bundles* can contain other *bundles* and *subproducts* can contain other *subproducts*.

- The `\*` software specification selects all products. Use this specification with caution.

The **version** component has the form:

> [**,r** <*op*> *revision*][**,a** <*op*> *arch*][**,v** <*op*> *vendor*]
> [**,c** <*op*> *category*][**,q=***qualifier*][**,l=***location*]
> [**,fr** <*op*> *revision*][**,fa** <*op*> *arch*]

- *location* applies only to installed software and refers to software installed to a location other than the default product directory.
- **fr** and **fa** apply only to filesets.
- The <*op*> (relational operator) component can be of the form:

  =, **==**, **>=**, **<=**, **<**, **>**, or **!=**

  which performs individual comparisons on dot-separated fields.

  For example, **r>=B.10.00** chooses all revisions greater than or equal to **B.10.00**. The system compares each dot-separated field to find matches.
- The **=** (equals) relational operator lets you specify selections with the shell wildcard and pattern-matching notations:

  [ ], *****, **?**, **!**

  For example, the expression **r=1[01].*** returns any revision in version 10 or version 11.
- All version components are repeatable within a single specification (e.g. **r>=A.12**, **r<A.20**). If multiple components are used, the selection must match all components.
- Fully qualified software specs include the **r=**, **a=**, and **v=** version components even if they contain empty strings. For installed software, **l=** is also included.
- No space or tab characters are allowed in a software selection.
- The software *instance_id* can take the place of the version component. It has the form:

  [*instance_id*]

  within the context of an exported catalog, where *instance_id* is an integer that distinguishes versions of products and bundles with the same tag.

Target Selections

The **swpackage** command supports the following syntax for a *target_selection*:

> **@ /***path*

If creating a distribution directory, this option defines the *path* to the directory. If creating a distribution tape, this option defines the *path* to the device file on which to write the distribution. When creating a distribution tape, the tape device (file) must exist, and the **-x media_type=tape** option must be specified (see below).

S

EXTERNAL INFLUENCES

Default Options

In addition to the standard options, several SD behaviors and policy options can be changed by editing the default values found in:

/var/adm/sw/defaults	the system-wide default values.
$HOME/.swdefaults	the user-specific default values.

Values must be specified in the defaults file using this syntax:

> [*command_name***.**]*option***=***value*

The optional *command_name* prefix denotes one of the SD commands.

You can also override default values from the command line with the **-x** or **-X** options:

> *command* **-x** *option***=***value*
>
> *command* **-X** *option_file*

The following section lists all of the keywords supported by **swpackage** and **swcopy**. If a default value exists, it is listed after the "=". The commands that this option applies to are also specified.

admin_directory=/var/adm/sw (for normal mode)
admin_directory=/var/home/LOGNAME/sw (for nonprivileged mode)

The location for SD logfiles and the default parent directory for the installed software catalog. The default value is **/var/adm/sw** for normal SD operations. When SD operates in nonprivileged mode (that is, when the **run_as_superuser** default option is set to **true**):

- The default value is forced to **/var/home/LOGNAME/sw**.
- The path element **LOGNAME** is replaced with the name of the invoking user, which SD reads from the system password file.
- If you set the value of this option to **HOME/***path*, SD replaces **HOME** with the invoking user's home directory (from the system password file) and resolves *path* relative to that directory. For example, **HOME/my_admin** resolves to the **my_admin** directory in your home directory.

SD's nonprivileged mode is intended only for managing applications that are specially designed and packaged. This mode cannot be used to manage the HP-UX operating system or patches to it. For a full explanation of nonprivileged SD, see the *Software Distributor Administration Guide*, available at the **http://docs.hp.com** web site.

See also the **run_as_superuser** option.

allow_partial_bundles=true

Determines whether to process partial bundles without WARNINGs and NOTEs. In the default state of **true**, this option tells **swpackage** to package what is available in the PSF. Missing or ambiguous bundle contents are ignored and no WARNINGs and NOTEs are issued.

When set to **false**, this option tells **swpackage** to expect all the bundle contents to be present and unique in the PSF. Objects that are ambiguous or missing generates a NOTE and every bundle with missing or ambiguous content generates a WARNING. (Note that **swpackage** succeeds even if NOTEs and WARNINGS occur.)

compress_cmd=/usr/contrib/bin/gzip

Defines the command called to compress files before installing, copying or packaging. If the **compression_type** option is set to other than **gzip** or **compress,** this path must be changed.

compress_files=false

If set to **true**, uncompressed files are compressed before transfer from a source. This enhances performance on slower networks for **swcopy** and **swinstall**, and results in smaller depots for **swcopy** and **swpackage**, unless the **uncompress_files** option is also set to **true**.

compress_index=false

Determines whether SD commands create compressed INDEX and INFO catalog files when writing to target depots or roots. The default of **false** does not create compressed files. When set to **true**, SD creates compressed and uncompressed INDEX and INFO files. The compressed files are named INDEX.gz and INFO.gz, and reside in the same directories as the uncompressed files.

Compressed files can enhance performance on slower networks, although they may increase disk space usage due to a larger Installed Products Database and depot catalog. SD controllers and target agents for HP-UX 11.01 and higher automatically load the compressed INDEX and INFO files from the source agent when:

- The source agent supports this feature.
- INDEX.gz or INFO.gz exist on the source depot.
- INDEX.gz or INFO.gz are not older than the corresponding uncompressed INDEX or INFO files.

The uncompressed INDEX or INFO file is accessed by the source agent if any problem occurs when accessing, transferring, or uncompressing the INDEX.gz or INFO.gz file.

compression_type=gzip

Defines the default compression type used by the agent when it compresses files during or after transmission. If **uncompress_files** is set to false, the **compression_type** is recorded for each file compressed so that the correct uncompression can later be applied during a **swinstall**, or a **swcopy** with **uncompress_files** set to true. The **compress_cmd** specified must produce files with the **compression_type** specified. The **uncompress_cmd** must be able to process files of the **compression_type** specified unless the format is **gzip**, which is uncompressed by the internal uncompressor (**funzip**).

create_target_acls=true

If creating a target depot, **swpackage** will create Access Control Lists (ACLs) for the depot (if it is new) and all products being packaged into it. If set to **false**, and if the user is the superuser, **swpackage** will not create ACLs. (The **swpackage** command never creates ACLs when software is packaged on to a distribution tape.)

distribution_source_directory=/var/spool/sw

Defines the default location of the source depot (when the **source_type** is *directory*). You can also use the *host***:***path* syntax. The **-s** option overrides this default.

distribution_target_directory=/var/spool/sw

Defines the default distribution directory of the target depot. The *target_selection* operand overrides this default.

distribution_target_serial=/dev/rmt/0m

Defines the default location of the target tape device file. The *target_selection* operand overrides this default.

enforce_dsa=true

Prevents a command from proceeding past the analysis phase if the disk space required is beyond the available free space of the impacted file systems. If set to **false**, then the install, copy, or package operation will use the file systems' minfree space and may fail because it reaches the file system's absolute limit.

follow_symlinks=false

Do not follow symbolic links in the package source files, but include the symbolic links in the packaged products. A value of **true** for this keyword causes **swpackage** to follow symbolic links in the package source files and include the files they reference in the packaged products.

include_file_revisions=false

Do not include each source file's revision attribute in the products being packaged. Because this operation is time consuming, by default the revision attributes are not included. If set to **true**, **swpackage** will execute *what*(1) and possibly *ident*(1) (in that order) to try to determine a file's revision attribute.

S

layout_version=1.0

Specifies the POSIX **layout_version** to which the SD commands conform when writing distributions and **swlist** output. Supported values are "1.0" (default) and "0.8".

SD object and attribute syntax conforms to the *layout_version 1.0* specification of the *IEEE POSIX 1387.2 Software Administration* standard. SD commands still accept the keyword names associated with the older layout version, but you should use **layout_version=0.8** only to create distributions readable by older versions of SD.

See the description of the **layout_version** option in *sd(5)* for more information.

log_msgid=0

Adds numeric identification numbers at the beginning of SD logfile messages:

0 (default) No identifiers are attached to messages.
1 Adds identifiers to ERROR messages only.
2 Adds identifiers to ERROR and WARNING messages.
3 Adds identifiers to ERROR, WARNING, and NOTE messages.
4 Adds identifiers to ERROR, WARNING, NOTE, and certain other informational messages.

logdetail=false

The **logdetail** option controls the amount of detail written to the log file. When set to

true, this option adds detailed task information (such as options specified, progress statements, and additional summary information) to the log file. This information is in addition to log information controlled by the **loglevel** option.

logfile=/var/adm/sw/sw<package>.log
Defines the default log file for the swpackage command.

loglevel=1
Controls the log level for the events logged to the command logfile, the target agent logfile, and the source agent logfile. This information is in addition to the detail controlled by the **logdetail** option. See **logdetail** for more information.
A value of
0 provides no information to the log files.
1 enables verbose logging to the log files.
2 enables very verbose logging to the log files.

media_capacity=1330
If creating a distribution tape or multiple-directory media such as a CD-ROM, this keyword specifies the capacity of the tape in one million byte units (not Mbytes). This option is required if the media is not a DDS tape or a disk file. Without this option, **swpackage** sets the size to the default of 1,330 Mbytes for tape or to the amount of free space on the disk up to **minfree** for a disk file. SD uses the same format across multiple directory media as it does for multiple serial media, including calculations of the correct size based partitioning of filesets and setting of the **media_sequence_number** attributes.

media_type=directory
Defines the type of distribution to create. The recognized types are **directory** and **tape**.

package_in_place=false
If set to **true**, **swpackage** does not put the files that make up a product in the target depot. Instead, **swpackage** inserts references to the original source files, saving disk space.

reinstall_files=false
Controls the overwriting of files, which may enhance performance on slow networks or disks. At the default value of false, SD compares each file in a source fileset to corresponding files on the target system. SD compares the files based on size, timestamp, and (optionally) the *checksum* (see **reinstall_files_use_cksum**). If the files are identical the files on the target system are not overwritten.

When set to true, SD does not compare files and overwrites any identical files on the target.

reinstall_files_use_cksum=false
Controls the use of checksum comparisons when the **reinstall_files** option is set to false. At the default value of true, this option causes SD to compute and compare checksums to determine if a new file should overwrite an old file. Use of checksums slows the comparison but is a more robust check for equivalency than size and time stamp.

If set to false, SD does not compute checksums and compares files only by size and timestamp.

run_as_superuser=true
This option controls SD's nonprivileged mode. This option is ignored (treated as true) when the invoking user is super-user.

When set to the default value of true, SD operations are performed normally, with permissions for operations either granted to a local super-user or set by SD ACLs. (See *swacl*(1M) for details on ACLs.)

When set to false and the invoking user is local and is *not* super-user, nonprivileged mode is invoked:

- Permissions for operations are based on the user's file system permissions.
- SD ACLs are ignored.
- Files created by SD have the uid and gid of the invoking user, and the mode of created files is set according to the invoking user's umask.

SD's nonprivileged mode is intended only for managing applications that are specially designed and packaged. This mode cannot be used to manage the HP-UX operating system or patches to it. For a full explanation of nonprivileged SD, see the *Software Distributor Administration Guide*, available at the **http://docs.hp.com** web site.

See also the **admin_directory** option.

software=
Defines the default *software_selections*. There is no supplied default. If there is more than one software selection, they must be separated by spaces. Software is usually specified in a software input file, as operands on the command line, or in the GUI.

source_file=psf
Defines the default location of the source product specification file (PSF). The **host:path** syntax is not allowed, only a valid **path** can be specified. The **-s** option overrides this value.

source_type=directory
Defines the default source type: **cdrom**, **file**, **directory**, or **tape**. The source type derived from the **-s** option overrides this value.

targets=
Defines the default *target_selections*. There is no supplied default. If there is more than one target selection, they must be separated by spaces. Targets are usually specified in a target input file, as operands on the command line, or in the GUI.

uncompress_cmd=
Defines the command to uncompress files when installing, copying, or packaging. This command processes files which were stored on the media in a compressed format. If the **compression_type** of the file is **gzip** then the internal uncompression (**funzip**) is used instead of the external **uncompress_cmd**.

verbose=
Controls the verbosity of a non-interactive command's output:
0 disables output to stdout. (Error and warning messages are always written to stderr).
1 enables verbose messaging to stdout.
2 for **swpackage** and **swmodify**, enables very verbose messaging to stdout.

The **-v** option overrides this default if it is set to 0. Applies to all commands.

write_remote_files=false
Prevents file operations on remote (NFS) file systems. All files destined for packaging on targets on a remote (NFS) file systems are skipped.

If set to true and if the superuser has write permission on the remote file system, the remote files are not skipped.

S

Session File

Each invocation of the **swpackage** command defines a packaging session. The invocation options, source information, software selections, and target hosts are saved before the installation or copy task actually commences. This lets you re-execute the command even if the session ends before proper completion.

Each session is saved to the file **$HOME/.sw/sessions/swpackage.last**. This file is overwritten by each invocation of **swpackage**.

You can also save session information to a specific file by executing **swpackage** with the **-C** *session__file* option.

A session file uses the same syntax as the defaults files. You can specify an absolute path for the session file. If you do not specify a directory, the default location for a session file is **$HOME/.sw/sessions/**.

To re-execute a session file, specify the session file as the argument for the **-S** *session__file* option of **swpackage**.

Note that when you re-execute a session file, the values in the session file take precedence over values in the system defaults file. Likewise, any command line options or parameters that you specify when you invoke **swpackage** take precedence over the values in the session file.

Environment Variables

The environment variable that affects **swpackage** is:

LANG Determines the language in which messages are displayed. If LANG is not specified or is set to the empty string, a default value of **C** is used. See the *lang*(5) man page by typing **man 5 lang** for more information.

NOTE: The language in which the SD agent and daemon log messages are displayed is set by the system configuration variable script, **/etc/rc.config.d/LANG**. For example, **/etc/rc.config.d/LANG**, must be set to **LANG=ja_JP.SJIS** or **LANG=ja_JP.eucJP** to make the agent and daemon log messages display in Japanese.

LC_ALL Determines the locale to be used to override any values for locale categories specified by the settings of **LANG** or any environment variables beginning with **LC_**.

LC_CTYPE
Determines the interpretation of sequences of bytes of text data as characters (e.g., single- versus multibyte characters in values for vendor-defined attributes).

LC_MESSAGES
Determines the language in which messages should be written.

LC_TIME
Determines the format of dates (*create_date* and *mod_date*) when displayed by **swlist**. Used by all utilities when displaying dates and times in **stdout**, **stderr**, and **logging**.

TZ Determines the time zone for use when displaying dates and times.

Signals

The **swpackage** command catches the signals SIGQUIT and SIGINT. If these signals are received, the command prints a message, sends a Remote Procedure Call (RPC) to the agents to wrap up, and then exits.

The agent ignores SIGHUP, SIGINT, and SIGQUIT. It immediately exits gracefully after receiving SIGTERM, SIGUSR1, or SIGUSR2. Killing the agent may leave corrupt software on the system, and thus should only be done if absolutely necessary. Note that when an SD command is killed, the agent does not terminate until completing the task in progress.

The daemon ignores SIGHUP, SIGINT and SIGQUIT. It immediately exits gracefully after receiving SIGTERM and SIGUSR2. After receiving SIGUSR1, it waits for completion of a copy or remove from a depot session before exiting, so that it can register or unregister depots if necessary. Requests to start new sessions are refused during this wait.

Locking

SD commands use a common locking mechanism for reading and modifying both root directories and software depots. This mechanism allows multiple readers but only one writer on a root or depot.

The SD commands which modify software in an (alternate) root directory are restricted from simultaneous modification using *fcntl*(2) locking on the file

var/adm/sw/products/swlock

relative to the root directory (e.g. **/var/adm/sw/products/swlock**).

The SD commands which modify software in a depot are restricted from simultaneous modification using *fcntl*(2) locking on the file

catalog/swlock

relative to the depot directory (e.g. **/var/spool/sw/catalog/swlock**).

All commands set *fcntl(2)* read locks on roots and depots using the **swlock** file mentioned above. When a read lock is set, it prevents other SD commands from performing modifications (i.e. from setting write locks).

PRODUCT SPECIFICATION FILE

This section summarizes the *product_specification_file* (PSF) which drives the **swpackage** session. See *swpackage*(4) for a detailed description of PSF syntax and semantics.

A PSF is structured as follows:

[*depot specification*]
  [*vendor specification*]

```
[category specification]
[bundle specification]
[product specification]
    [control script specification]
    [subproduct specification]
    [fileset specification]
        [control script specification]
        [file specification]
    [fileset specification]
    ...
[product specification]
...
```

If errors encountered while parsing the PSF result in no valid product definitions, **swpackage** terminates. All errors are logged to both stderr and the logfile. In summary, the **swpackage** user can:

- Specify one or more products;
- For each product, specify one or more filesets.
- For each fileset, specify one or more files.
- (optional) Specify attributes for the target depot/tape;
- (optional) Specify one or more bundles, defining the bundle contents;
- (optional) Specify vendor information for products and bundles;
- (optional) Specify category information for products, bundles and patches.
- (optional) For each product, specify one or more subproducts, defining the subproduct contents;
- (optional) For each product or fileset, specify one or more control scripts.

RETURN VALUES

The **swpackage** command returns:

0 The products specified in the *product_specification_file* were successfully packaged into the target depot/tape.

1 An error occurred during the **swpackage** session (e.g. bad syntax in the *product_specification_file*.) Review stderr or the log file for details.

DIAGNOSTICS

The **swpackage** command writes to stdout, stderr, and to the logfile.

Standard Output

The **swpackage** command writes messages for significant events. These include:

- a begin and end session message,
- selection, analysis, packaging, and tape creation messages.

Standard Error

The **swpackage** command writes messages for all WARNING and ERROR conditions to stderr.

Logfile

The **swpackage** command logs detailed events to the log file **/var/adm/sw/swpackage.log**. The user can specify a different logfile by modifying the **logfile** option.

EXAMPLES

Package the products defined in the PSF *products* into the default target depot:

```
swpackage -s products
```

Preview the same operation (do not create the target depot), and generate very verbose output:

```
swpackage -p -vv -s products
```

Package the products into the target depot *no_files*, insert references to the source files instead of copying them into the depot:

```
swpackage -s products -x package_in_place=true  @ no_files
```

Re-package a specific fileset:

```
swpackage -s products -x package_in_place=true product.fileset  @ no_files
```

Re-package the entire contents of the depot **/var/spool/sw** onto the tape at **/dev/rmt/0m**:

```
swpackage -s /var/spool/sw -x media_type=tape  @ /dev/rmt/0m
```

FILES

/dev/rmt/0m
The default location of a source and target tape. (Note that SD can read both **tar** and **cpio** tape depots.)

$HOME/.swdefaults
Contains the user-specific default values for some or all SD options.

$HOME/.sw/sessions/
Contains session files automatically saved by the SD commands, or explicitly saved by the user.

/usr/lib/sw/sys.defaults
Contains the master list of current SD options with their default values.

/var/adm/sw/
The directory which contains all of the configurable and non-configurable data for SD. This directory is also the default location of logfiles.

/var/adm/sw/defaults
Contains the active system-wide default values for some or all SD options.

/var/spool/sw/
The default location of a source and target software depot.

AUTHOR

swpackage was developed by the Hewlett-Packard Company and Mark H. Colburn (see *pax*(1)).

SEE ALSO

sd(4), swpackage(4), sd(5).

Software Distributor Administration Guide, available at **http://docs.hp.com**.

SD customer web site at **http://software.hp.com/SD_AT_HP/**.

S

NAME

swreg - register or unregister depots and roots

SYNOPSIS

swreg -l *level* [**-u**] [**-v**] [**-C** *session_file*] [**-f** *object_file*] [**-S** *session_file*] [**-t** *target_file*] [**-x** *option=value*] [**-X** *option_file*] [*objects_to_(un)register*] [**@** *target_selections*]

Remarks

- This command supports operations on remote systems. See **Remote Operation** below.
- For an overview of all SD commands, see the *sd*(5) man page by typing **man 5 sd** on the command line.

DESCRIPTION

The **swreg** command controls the visibility of depots and roots to users who are performing software management tasks. It must be used to register depots created by **swpackage**.

By default, the **swcopy** command registers newly created depots. By default, the **swinstall** command registers newly created alternate roots (the root, "/", is not automatically registered). The **swremove** command unregisters a depot, or root, when or if the depot is empty. The user invokes **swreg** to explicitly (un)register a depot when the automatic behaviors of **swcopy**, **swinstall**, **swpackage**, and **swremove** do not suffice. For example:

- Making a CD-ROM or other removable media available as a registered depot.
- Registering a depot created directly by **swpackage**.
- Unregistering a depot without removing it with **swremove**.

Remote Operation

You can enable SD to manage software on remote systems. To let the root user from a central SD *controller* (also called the *central management server* or *manager node*) perform operations on a remote *target* (also called the *host* or *agent*):

1) Install a special HP ServiceControl Manager fileset on the remote systems. This enables remote operations by automatically setting up the root, host, and template Access Control Lists (ACLs) on the remote machines and permitting root access from the controller system. To install the fileset, run the following command on each remote system:

swinstall -s *controller* **:/var/opt/mx/depot11 AgentConfig.SD-CONFIG**

NOTES:

- *controller* is the name of the central management server.
- If the target is running HP-UX 10.20, use the same command but substitute **depot10** for **depot11**.
- Targets previously set up by SD/OV to be managed by this controller do not need this step.
- SD does not require any other ServiceControl Manager filesets.

2) (Optional) **swinstall**, **swcopy**, and **swremove** have enhanced GUI interfaces for remote operations. Enable the enhanced GUIs by creating the **.sdkey** file on the controller. Use this command:

```
touch /var/adm/sw/.sdkey
```

NOTES:

- This step is not required when you use SD from within the HP ServiceControl Manager.
- See *sd(5)*, *swinstall(1M)*, *swcopy(1M)*, *swjob(1M)*, *swlist(1M)*, or *swremove(1M)* for more information on interactive operations.

NOTE: You can also set up remote access by using **swacl** directly on the remote machines to grant root or non-root access to users from the controller system.

Options

The **swreg** command supports the following options:

-l *level* Specify the *level* of the object to register or unregister. Exactly one level must be specified. The supported levels are:

depot	The object to be registered is a depot.
root	The object to be registered is a root.
shroot	The object to register is a shared root (HP-UX 10.X only).
prroot	The object to register is a private root (HP-UX 10.X only).

-u Causes **swreg** to unregister the specified objects instead of registering them.

-v Turns on verbose output to stdout. (The **swreg** logfile is not affected by this option.) Verbose output is enabled by default, see the **verbose** option below.

-C *session_file*
Save the current options and operands to *session_file*. You can enter a relative or absolute path with the file name. The default directory for session files is **$HOME/.sw/sessions/**. You can recall a session file with the **-S** option.

-f *object_file* Read the list of depot or root objects to register or unregister from *object_file* instead of (or in addition to) the command line.

-S *session_file*
Execute **swreg** based on the options and operands saved from a previous session, as defined in *session_file*. You can save session information to a file with the **-C** option.

-t *target_file* Read the list of target *hosts* on which to register the depot or root objects from *target_file* instead of (or in addition to) the command line.

-x *option=value*
Set the session *option* to *value* and override the default value (or a value in an alternate *option_file* specified with the **-X** option). Multiple **-x** options can be specified.

-X *option_file* Read the session options and behaviors from *option_file*.

Operands

The **swreg** command supports the following syntax for each *object_to_register*:

path

Each operand specifies an object to be registered or unregistered.

The **swreg** command supports the following syntax for each *target_selection*:

[*host*]

EXTERNAL INFLUENCES

Default Options

In addition to the standard options, several SD behaviors and policy options can be changed by editing the default values found in:

/var/adm/sw/defaults	the system-wide default values.
$HOME/.swdefaults	the user-specific default values.

Values must be specified in the defaults file using this syntax:

[*command_name*.]*option*=*value*

The optional *command_name* prefix denotes one of the SD commands. Using the prefix limits the change in the default value to that command. If you leave the prefix off, the change applies to all commands.

You can also override default values from the command line with the **-x** or **-X** options:

command **-x** *option*=*value*

command **-X** *option_file*

The following list describes keywords supported by the **swreg** command. If a default value exists, it is listed after the "=".

admin_directory=/var/adm/sw (for normal mode)
admin_directory=/var/home/LOGNAME/sw (for nonprivileged mode)
The location for SD logfiles and the default parent directory for the installed software catalog. The default value is **/var/adm/sw** for normal SD operations. When SD operates in nonprivileged mode (that is, when the **run_as_superuser** default option is set to **true**):

S

- The default value is forced to **/var/home/LOGNAME/sw**.
- The path element **LOGNAME** is replaced with the name of the invoking user, which SD reads from the system password file.
- If you set the value of this option to **HOME/***path*, SD replaces **HOME** with the invoking user's home directory (from the system password file) and resolves *path* relative to that directory. For example, **HOME/my_admin** resolves to the **my_admin** directory in your home directory.

SD's nonprivileged mode is intended only for managing applications that are specially designed and packaged. This mode cannot be used to manage the HP-UX operating system or patches to it. For a full explanation of nonprivileged SD, see the *Software Distributor Administration Guide*, available at the **http://docs.hp.com** web site.

See also the **run_as_superuser** option.

distribution_target_directory=/var/spool/sw
Defines the location of the depot object to register if no objects are specified and the **-l** option is specified.

level= Defines the default level of objects to register or unregister. The valid levels are:

depot	Depots which exist at the specified target hosts.
root	All alternate roots.
shroot	All registered shared roots *(HP-UX 10.X only)*.
prroot	All registered private roots *(HP-UX 10.X only)*.

log_msgid=0
Adds numeric identification numbers at the beginning of SD logfile messages:
0 (default) No identifiers are attached to messages.
1 Adds identifiers to ERROR messages only.
2 Adds identifiers to ERROR and WARNING messages.
3 Adds identifiers to ERROR, WARNING, and NOTE messages.
4 Adds identifiers to ERROR, WARNING, NOTE, and certain other informational messages.

logfile=/var/adm/sw/swreg.log
Specifies the default command log file for the **swreg** command.

logdetail=false[true]
The **logdetail** option controls the amount of detail written to the log file. When set to **true**, this option adds detailed task information (such as options specified, progress statements, and additional summary information) to the log file. This information is in addition to log information controlled by the **loglevel** option. See the *sd*(5) man page for additional information by typing **man 5 sd**.

S

loglevel=1
Controls the log level for the events logged to the command logfile, the target agent logfile, and the source agent logfile. This information is in addition to the detail controlled by the **logdetail** option. (See also **logdetail**.) A value of
0 provides no information to the logfile.
1 enables verbose logging to the logfiles.
2 enables very verbose logging to the logfiles.

objects_to_register=
Defines the default objects to register or unregister. There is no supplied default (see **distribution_target_directory** above). If there is more than one object, they must be separated by spaces.

rpc_binding_info=ncacn_ip_tcp:[2121] ncadg_ip_udp:[2121]
Defines the protocol sequence(s) and endpoint(s) on which the daemon listens and which the other commands use to contact the daemon. If the connection fails for one protocol sequence, the next is attempted. SD supports both the tcp (ncacn_ip_tcp:[2121]) and udp (ncadg_ip_udp:[2121]) protocol sequence on most platforms.

See the *sd*(5) man page by typing **man 5 sd** for details on specifying this option.

rpc_timeout=5
Relative length of the communications timeout. This is a value in the range from 0 to 9 and

is interpreted by the DCE RPC. Higher values mean longer times; you may need a higher value for a slow or busy network. Lower values will give faster recognition on attempts to contact hosts that are not up or are not running **swagentd**. Each value is approximately twice as long as the preceding value. A value of 5 is about 30 seconds for the **ncadg_ip_udp** protocol sequence. This option may not have any noticeable impact when using the **ncacn_ip_tcp** protocol sequence.

run_as_superuser=true

This option controls SD's nonprivileged mode. This option is ignored (treated as true) when the invoking user is super-user.

When set to the default value of true, SD operations are performed normally, with permissions for operations either granted to a local super-user or set by SD ACLs. (See *swacl*(1M) for details on ACLs.)

When set to false and the invoking user is local and is *not* super-user, nonprivileged mode is invoked:

- Permissions for operations are based on the user's file system permissions.
- SD ACLs are ignored.
- Files created by SD have the uid and gid of the invoking user, and the mode of created files is set according to the invoking user's umask.

SD's nonprivileged mode is intended only for managing applications that are specially designed and packaged. This mode cannot be used to manage the HP-UX operating system or patches to it. For a full explanation of nonprivileged SD, see the *Software Distributor Administration Guide*, available at the **http://docs.hp.com** web site.

See also the **admin_directory** option.

select_local=true

If no *target_selections* are specified, select the default **distribution_target_directory** of the local host as the *target_selection* for the command.

targets=

Defines the default *target* hosts on which to register or unregister the specified root or depot objects. There is no supplied default (see **select_local** above). If there is more than one target selection, they must be separated by spaces.

verbose=1

Controls the verbosity of the **swreg** output (stdout). A value of

0 disables output to stdout. (Error and warning messages are always written to stderr).
1 enables verbose messaging to stdout.

S

Session File

Each invocation of the **swreg** command defines a registration session. The invocation options, source information, software selections, and target hosts are saved before the installation or copy task actually commences. This lets you re-execute the command even if the session ends before proper completion.

Each session is saved to the file **$HOME/.sw/sessions/swreg.last**. This file is overwritten by each invocation of **swreg**.

You can also save session information to a specific file by executing **swreg** with the **-C** *session__file* option.

A session file uses the same syntax as the defaults files. You can specify an absolute path for the session file. If you do not specify a directory, the default location for a session file is **$HOME/.sw/sessions/**.

To re-execute a session file, specify the session file as the argument for the **-S** *session__file* option of **swreg**.

Note that when you re-execute a session file, the values in the session file take precedence over values in the system defaults file. Likewise, any command line options or parameters that you specify when you invoke **swreg** take precedence over the values in the session file.

Environment Variables

SD programs are affected by external environment variables.

SD programs that execute control scripts set environment variables for use by the control scripts.

In addition, **swinstall** sets environment variables for use when updating the HP-UX operating system and modifying the HP-UX configuration.

The environment variable that affects the **swreg** command is:

LANG Determines the language in which messages are displayed. If LANG is not specified or is set to the empty string, a default value of **C** is used. See the *lang*(5) man pages by typing **man 5 lang** for more information.

NOTE: The language in which the SD agent and daemon log messages are displayed is set by the system configuration variable script, **/etc/rc.config.d/LANG**. For example, **/etc/rc.config.d/LANG**, must be set to **LANG=ja_JP.SJIS** or **LANG=ja_JP.eucJP** to make the agent and daemon log messages display in Japanese.

LC_ALL Determines the locale to be used to override any values for locale categories specified by the settings of **LANG** or any environment variables beginning with **LC_**.

LC_CTYPE
Determines the interpretation of sequences of bytes of text data as characters (e.g., single- versus multibyte characters in values for vendor-defined attributes).

LC_MESSAGES
Determines the language in which messages should be written.

LC_TIME
Determines the format of dates (*create_date* and *mod_date*) when displayed by **swlist**. Used by all utilities when displaying dates and times in **stdout**, **stderr**, and **logging**.

TZ Determines the time zone for use when displaying dates and times.

Signals

The **swreg** command catches the signals SIGQUIT and SIGINT. If these signals are received, **swreg** prints a message, sends a Remote Procedure Call (RPC) to the daemons to wrap up, and then exits.

RETURN VALUES

The **swreg** command returns:

0 The *objects_to_register* were successfully (un)registered.
1 The register or unregister operation failed on all *target_selections*.
2 The register or unregister operation failed on some *target_selections*.

DIAGNOSTICS

S

The **swreg** command writes to stdout, stderr, and to the daemon logfile.

Standard Output

The **swreg** command writes messages for significant events. These include:

- a begin and end session message,
- selection and execution task messages for each *target_selection*.

Standard Error

The **swreg** command writes messages for all WARNING and ERROR conditions to stderr.

Logging

The **swreg** command logs summary events at the host where the command was invoked. It logs events about each (un)register operation to the **swagentd** logfile associated with each *target_selection*.

EXAMPLES

Create a new depot with **swpackage**, then register it with **swreg**:

```
swpackage -s psf -d /var/spool/sw
swreg -l depot  /var/spool/sw
```

Unregister the default depot at several hosts:

```
swreg -u -l depot /var/spool/sw @ hostA hostB hostC
```

Unregister a specific depot at the local host:

```
swreg -u -l depot /cdrom
```

FILES

`$HOME/.swdefaults`
Contains the user-specific default values for some or all SD options.

`/usr/lib/sw/sys.defaults`
Contains the master list of current SD options with their default values.

`/var/adm/sw/`
The directory which contains all of the configurable and non-configurable data for SD. This directory is also the default location of logfiles.

`/var/adm/sw/defaults`
Contains the active system-wide default values for some or all SD options.

`/var/adm/sw/host_object`
The file which stores the list of depots registered at the local host.

AUTHOR

`swreg` was developed by the Hewlett-Packard Company.

SEE ALSO

swacl(1M), swagentd(1M), swask(1M), swconfig(1M), swcopy(1M), swinstall(1M), swjob(1M), swlist(1M), swmodify(1M), swpackage(1M), swremove(1M), swverify(1M), install-sd(1M), sd(4), swpackage(4), sd(5),

Software Distributor Administration Guide, available at **`http://docs.hp.com`**.

SD customer web site at **`http://software.hp.com/SD_AT_HP/`**.

NAME

swremove - unconfigure and remove software products

SYNOPSIS

swremove [*XToolkit Options*] [**-d**|**-r**] [**-i**] [**-p**] [**-v**] [**-C** *session_file*] [**-f** *software_file*] [**-J** *jobid*] [**-Q** *date*] [**-S** *session_file*] [**-t** *target_file*] [**-x** *option=value*] [**-X** *option_file*] [*software_selections*] [**@** *target_selections*]

Remarks

- **swremove** supports an interactive user interface (GUI) that can be invoked alone or by the **sd** command. See **Interactive Operation** below.
- This command supports operations on remote systems. See **Remote Operation** below.
- For an overview of all SD commands, see the *sd*(5) man page by typing **man 5 sd** on the command line.

DESCRIPTION

The **swremove** command removes *software_selections* from *target_selections* (e.g. root file systems). When removing installed software, **swremove** also unconfigures the software before it is removed. The software is not unconfigured when removed from an alternate root directory since it was not configured during installation. When removing available software (within a depot), **swremove** also does not perform the unconfiguration task.

NOTE : Selecting a bundle for removal does not always remove all filesets in that bundle. If a particular fileset is required by another bundle, that fileset will not be removed. For example, if the bundles **Pascal** and **FORTRAN** both use the fileset *Debugger.Run* and you try to remove **FORTRAN**, the fileset *Debugger.Run* will not be removed because it is also used by the bundle **Pascal**. This prevents the removal of one bundle from inadvertently causing the removal of filesets needed by another bundle.

Remote Operation

You can enable Software Distributor (SD) to manage software on remote systems. To let the root user from a central SD *controller* (also called the *central management server* or *manager node*) perform operations on a remote *target* (also called the *host* or *agent*):

1) Install a special HP ServiceControl Manager fileset on the remote systems. This enables remote operations by automatically setting up the root, host, and template Access Control Lists (ACLs) on the remote machines and permitting root access from the controller system. To install the fileset, run the following command on each remote system:

swinstall -s *controller***:/var/opt/mx/depot11 AgentConfig.SD-CONFIG**

NOTES:

- *controller* is the name of the central management server.
- If the target is running HP-UX 10.20, use the same command but substitute **depot10** for **depot11**.
- Targets previously set up by SD/OV to be managed by this controller do not need this step.
- SD does not require any other ServiceControl Manager filesets.

2) (Optional) Enable the GUI interfaces for remote operations by creating the **.sdkey** file on the controller. Use this command:

```
touch /var/adm/sw/.sdkey
```

(This step is not required when you use SD from within the HP ServiceControl Manager.)

NOTE: You can also set up remote access by using the *swacl*(1M) command directly on the remote machines to grant root or non-root access to users from the controller system.

Interactive Operation

swremove supports a graphical user interface (GUI) or a terminal user interface (in which screen navigation is done with the keyboard and no mouse) if your terminal or display cannot support the GUI.

To invoke the GUI, type

swremove on the command line (without command-line arguments) or include **-i** with any other command-line options when you invoke **swremove** from the command line.

The **sd** command provides an interactive interface for monitoring software jobs. You can also use it to invoke the **swinstall**, **swcopy**, or **swremove** GUIs.

If you have enabled SD's central management features, **swinstall**, **swcopy**, and **swremove** provide enhanced GUIs to support operations on remote machines. See **Remote Operations** above.

Removing Patches or Patch Rollback Files

To remove patch software, rollback files corresponding to the patch *must* be available for rollback. You must remove the base software modified by the patch. (Removing the base software also removes the patches associated with that software.)

To commit (make permanent) a patch, use the **swmodify** command's **patch_commit** option to remove the files saved for patch rollback, or use the **swinstall** command's *save_patch_files* option to not save them initially. See *swmodify*(1M) and *swinstall*(1M) for more information.

Control Scripts

When removing installed software, the **swremove** command executes several vendor-supplied scripts (if they exist) during the removal of the *software_selections*. The **swremove** command supports the following scripts:

checkremove
a script executed during the analysis of each *target_selection*, it checks to make sure the removal can be attempted. If this check fails, the software product will not be removed.

preremove
a script executed immediately before the software files are removed.

postremove
a script executed immediately after the software files are removed.

unconfigure
a script executed during the unconfiguration of each *target_selection*, it unconfigures the host for the software (and the software for the host). The **preremove** and **postremove** scripts are not intended for unconfiguration tasks. They are to be used for simple file management needs such as restoring files moved during install. The **unconfigure** script allows the **swremove** command to unconfigure the hosts on which it has been running before removing the software specified.

Options

The **swremove** supports the following options:

XToolKit Options
The **swremove** command supports a subset of the standard X Toolkit options to control the appearance of the GUI. The supported options are: **-bg**, **-background**, **-fg**, **-foreground**, **-display**, **-name**, **-xrm**, and **-synchronous**. See the *X*(1) manual page for a definition of these options.

-d Operate on a depot rather than installed software.

-r Operates on an alternate root directory, which must be specified the @ *target_selections* option. Note that unconfigure scripts are not run when removing software from an alternate root directory. (This option is not required for alternate root operations but is maintained for backward compatibility. See the **Alternate Root Directory and Depot Directory** heading in *sd*(5) for more information.)

-i Runs the command in interactive mode (Graphical User Interface). See the **Interactive Operation** and **Remote Operation** headings above for additional details.

-p Previews a remove task by running the session through the analysis phase only.

-v Turns on verbose output to stdout. (The **swremove** log file is not affected by this option.) Verbose output is controlled by the default **verbose=x**.

S

-C *session_file*
Save the current options and operands to *session_file*. You can enter a relative or absolute path with the file name. The default directory for session files is **$HOME/.sw/sessions/**. You can recall a session file with the **-S** option.

-f *software_file*
Read the list of *software_selections* from *software_file* instead of (or in addition to) the command line.

-J *jobid* Executes a previously scheduled job. This is the syntax used by the daemon to start the job.

-Q *date* Schedules a job for the specified date. You can change the date format by modifying the file **/var/adm/sw/getdate.templ**.

-S *session_file*
Execute **swremove** based on the options and operands saved from a previous session, as defined in *session_file*. You can save session information to a file with the **-C** option.

-t *target_file* Read the list of *target_selections* from *target_file* instead of (or in addition to) the command line.

-x *option=value*
Set the session *option* to *value* and override the default value (or a value in an alternate *option_file* specified with the **-X** option). Multiple **-x** options can be specified.

-X *option_file* Read the session options and behaviors from *option_file*.

Operands

swremove supports two types of operands: *software selections* followed by *target selections*. These operands are separated by the "@" (at) character. This syntax implies that the command operates on "software selections at targets".

Software Selections

The *selections* operands consist of *software_selections*.

swremove supports the following syntax for each *software_selection*:

bundle[**.***product*[**.***subproduct*][**.***fileset*]][**,***version*]

product[**.***subproduct*][**.***fileset*][**,***version*]

- The **=** (equals) relational operator lets you specify selections with the following shell wildcard and pattern-matching notations:

  [ ], *****, **?**

  For example, the following expression removes all bundles and products with tags that end with "man":

  ```
  swremove sw_server *man
  ```

- *Bundles* and *subproducts* are recursive. *Bundles* can contain other *bundles* and *subproducts* can contain other *subproducts*. For example:

  ```
  swremove bun1.bun2.prod.sub1.sub2.fset,r=1.0
  ```

  or (using expressions):

  ```
  swremove bun[12].bun?.prod.sub*,a=HP-UX
  ```

- The **\*** software specification selects all products. Use this specification with caution.

The **version** component has the form:

[**,r** <*op*> *revision*][**,a** <*op*> *arch*][**,v** <*op*> *vendor*]
[**,c** <*op*> *category*][**,q=***qualifier*][**,l=***location*]
[**,fr** <*op*> *revision*][**,fa** <*op*> *arch*]

- *location* applies only to installed software and refers to software installed to a location other than the default product directory.

- **fr** and **fa** apply only to filesets.
- The *<op>* (relational operator) component can be of the form:

  =, **==**, **>=**, **<=**, **<**, **>**, or **!=**

  which performs individual comparisons on dot-separated fields.

  For example, **r>=B.10.00** chooses all revisions greater than or equal to **B.10.00**. The system compares each dot-separated field to find matches. Shell patterns are not allowed with these operators.
- The **=** (equals) relational operator lets you specify selections with the shell wildcard and pattern-matching notations:

  [ ], *****, **?**, **!**

  For example, the expression **r=1[01].*** returns any revision in version 10 or version 11.
- All version components are repeatable within a single specification (e.g. **r>=A.12**, **r<A.20**). If multiple components are used, the selection must match all components.
- Fully qualified software specs include the **r=**, **a=**, and **v=** version components even if they contain empty strings.
- No space or tab characters are allowed in a software selection.
- The software *instance_id* can take the place of the version component. It has the form:

  [*instance_id*]

  within the context of an exported catalog, where *instance_id* is an integer that distinguishes versions of products and bundles with the same tag.

Target Selections

swremove supports the following syntax for each *target_selection*:

[*host*][**:**][**/***directory*]

The **:** (colon) is required if both a host and directory are specified.

EXTERNAL INFLUENCES

Default Options

In addition to the standard options, you can change **swremove** behavior and policy options by editing the default values found in:

/var/adm/sw/defaults	the system-wide default values.
$HOME/.swdefaults	the user-specific default values.

Values must be specified in the defaults file using this syntax:

[*command_name***.**]*option***=***value*

The optional *command_name* prefix denotes one of the SD commands. Using the prefix limits the change in the default value to that command. If you leave the prefix off, the change applies to all commands.

You can also override default values from the command line with the **-x** or **-X** options:

command **-x** *option***=***value*

command **-X** *option_file*

The following section lists all of the keywords supported by **swremove**. If a default value exists, it is listed after the "=".

The policy options that apply to **swremove** are:

admin_directory=/var/adm/sw (for normal mode)
admin_directory=/var/home/LOGNAME/sw (for nonprivileged mode)

The location for SD logfiles and the default parent directory for the installed software catalog. The default value is **/var/adm/sw** for normal SD operations. When SD operates in nonprivileged mode (that is, when the **run_as_superuser** default option is set to **true**):

- The default value is forced to **/var/home/LOGNAME/sw**.

S

- The path element **LOGNAME** is replaced with the name of the invoking user, which SD reads from the system password file.
- If you set the value of this option to **HOME/***path*, SD replaces **HOME** with the invoking user's home directory (from the system password file) and resolves *path* relative to that directory. For example, **HOME/my_admin** resolves to the **my_admin** directory in your home directory.
- If you set the value of the **installed_software_catalog** default option to a relative path, that path is resolved relative to the value of this option.

SD's nonprivileged mode is intended only for managing applications that are specially designed and packaged. This mode cannot be used to manage the HP-UX operating system or patches to it. For a full explanation of nonprivileged SD, see the *Software Distributor Administration Guide*, available at the **http://docs.hp.com** web site.

See also the **installed_software_catalog** and **run_as_superuser** options.

agent_auto_exit=true

Causes the target agent to automatically exit after Execute phase, or after a failed Analysis phase. This is forced to **false** when the controller is using an interactive user interface, or when **-p** (preview) is used. This enhances network reliability and performance. The default value of **true** causes the target agent to automatically exit when appropriate. When set to **false**, the target agent will not exit until the controller ends the session.

agent_timeout_minutes=10000

Causes a target agent to exit if it has been inactive for the specified time. This can be used to make target agents more quickly detect lost network connections since RPC can take as long as 130 minutes to detect a lost connection. The recommended value is the longest period of inactivity expected in your environment. For command line invocation, a value between 10 minutes and 60 minutes is suitable. A value of 60 minutes or more is recommended when the GUI will be used. The default of 10000 is slightly less than 7 days.

allow_split_patches=false

Permits the use of single patch filesets without "sibling" filesets. In the default state of **false**, removal of a single fileset from a multi-fileset patch automatically includes any other fileset that are part of the patch, based on the ancestor filesets of the target fileset. (This behavior applies to filesets selected directly by the user and to filesets automatically selected by SD to resolve software dependencies.)

When set to **true**, SD allows a single patch fileset to be removed without including the sibling filesets. This allows a target to contain a patch that has been "split" into its component filesets. WARNING: Splitting a patch can create a situation in which one fileset in a sibling group would be removed by a patch, while the other filesets would not.

auto_kernel_build=true

Normally set to true. Specifies whether the removal of a kernel fileset should rebuild the kernel or not. If the kernel rebuild succeeds, the system automatically reboots. If set to false, the system continues to run the current kernel.

If the **auto_kernel_build** option is set to **true**, the **autoreboot** option must also be set to **true**. If the **auto_kernel_build** option is set to **false**, the value of the **autoreboot** option does not matter.

autoreboot=false

Prevents the removal of software requiring a reboot from the non-interactive interface. If set to **true**, then this software can be removed and the target system(s) will be automatically rebooted.

An interactive session always asks for confirmation before software requiring a reboot is removed.

If the **auto_kernel_build** option is set to **true**, the **autoreboot** option must also be set to **true**. If the **auto_kernel_build** option is set to **false**, the value of the **autoreboot** option does not matter.

autoremove_job=false

Controls automatic job removal. If the job is automatically removed, job information (job status or controller/agent log files) cannot be queried with **swjob**.

autoselect_dependents=false
Automatically selects all software that depends on the specified software. When set to **true**, and any software that other software depends on is selected for removal, **swremove** automatically selects that other software. If set to **false**, automatic selections are not made to resolve requisites.

autoselect_reference_bundles=true
If **true**, bundles that have the is_sticky attribute set to **true** will be automatically removed when the last of its contents is removed. If **false**, the sticky bundles will not be automatically removed.

compress_index=false
Determines whether SD commands create compressed INDEX and INFO catalog files when writing to target depots or roots. The default of **false** does not create compressed files. When set to **true**, SD creates compressed and uncompressed INDEX and INFO files. The compressed files are named INDEX.gz and INFO.gz, and reside in the same directories as the uncompressed files.

Compressed files can enhance performance on slower networks, although they may increase disk space usage due to a larger Installed Products Database and depot catalog. SD controllers and target agents for HP-UX 11.01 and higher automatically load the compressed INDEX and INFO files from the source agent when:

- The source agent supports this feature.
- INDEX.gz or INFO.gz exist on the source depot.
- INDEX.gz or INFO.gz are not older than the corresponding uncompressed INDEX or INFO files.

The uncompressed INDEX or INFO file is accessed by the source agent if any problem occurs when accessing, transferring, or uncompressing the INDEX.gz or INFO.gz file.

controller_source=
Specifies the location of a depot for the controller to access to resolve selections. Setting this option can reduce network traffic between the controller and the target. Use the target selection syntax to specify the location:

[*host*][**:**][*path*]

This option has no effect on which sources the target uses and is ignored when used with the Interactive User Interface.

distribution_target_directory=/var/spool/sw
Defines the default location of the target depot.

enforce_dependencies=true
Requires that all dependencies specified by the *software_selections* be resolved at the *target_selections*. For **swremove**, if a selected fileset has dependents (i.e. other software depends on the fileset) and they are not selected, do not remove the selected filesets. If set to **false**, dependencies will still be checked, but not enforced.

enforce_scripts=true
Controls the handling of errors generated by scripts. If **true**, and a script returns an error, the **swremove** operation halts. An error message appears reporting that the execution phase failed. If **false**, all script errors are treated as warnings, and **swremove** attempts to continue operation. A warning message appears reporting that the execution succeeded. The message wording identifies whether the failure occurred in the configure/unconfigure, checkremove, preremove, or postremove phases.

force_single_target=false
This option applies only to the Interactive User Interface when no SD-OV license is in effect on a system that is a diskless server. It causes **swremove** to run in a single target mode, even though a diskless server normally causes **swremove** to run in multi-target mode.

installed_software_catalog=products
Defines the directory path where the Installed Products Database (IPD) is stored. This information describes installed software. When set to an absolute path, this option defines the location of the IPD. When this option contains a relative path, the SD controller appends the value to the value specified by the **admin_directory** option to determine

the path to the IPD. For alternate roots, this path is resolved relative to the location of the alternate root. This option does not affect where software is installed, only the IPD location.

This option permits the simultaneous installation and removal of multiple software applications by multiple users or multiple processes, with each application or group of applications using a different IPD.

Caution: use a specific **installed_software_catalog** to manage a specific application. SD does not support multiple descriptions of the same application in multiple IPDs.

See also the **admin_directory** and **run_as_superuser** options, which control SD's nonprivileged mode. (This mode is intended only for managing applications that are specially designed and packaged. This mode cannot be used to manage the HP-UX operating system or patches to it. For a full explanation of nonprivileged SD, see the *Software Distributor Administration Guide*, available at the **http://docs.hp.com** web site.)

job_title=
Specifies an ASCII string giving a title to a job. It is displayed along with the job ID to provide additional identifying information about a job when **swjob** is invoked.

log_msgid=0
Adds numeric identification numbers at the beginning of SD logfile messages:
0 (default) No identifiers are attached to messages.
1 Adds identifiers to ERROR messages only.
2 Adds identifiers to ERROR and WARNING messages.
3 Adds identifiers to ERROR, WARNING, and NOTE messages.
4 Adds identifiers to ERROR, WARNING, NOTE, and certain other informational messages.

logdetail=false
Controls the amount of detail written to the log file. When set to **true**, this option adds detailed task information (such as options specified, progress statements, and additional summary information) to the log file. This information is in addition to log information controlled by the **loglevel** option.

See the **loglevel** option and the *sd(5)* manual page for more information.

logfile=/var/adm/sw/swremove.log
This is the default command log file for the **swremove** command.

loglevel=1
Controls the log level for the events logged to the command logfile, the target agent logfile, and the source agent logfile. This information is in addition to the detail controlled by the **logdetail** option.

0 provides no information to the logfile.
1 enables verbose logging to the log files.
2 enables very verbose logging to the log files.

See the **logdetail** option and the *sd*(5) manual page for more information.

mount_all_filesystems=true
By default, the **swremove** command attempts to automatically mount all filesystems in the **/etc/fstab** file at the beginning of the analysis phase, to ensure that all listed filesystems are mounted before proceeding. This policy helps to ensure that files which may be on mounted filesystems are available to be removed.

If set to **false**, the mount operation is not attempted, and no check of the current mounts is performed.

polling_interval=2
Defines the polling interval used by the Interactive UI of the controller. It specifies how often each target agent will be polled to obtain status information about the task being performed. When operating across wide-area networks, the polling interval can be increased to reduce network overhead.

remove_empty_depot=true
Controls whether a depot is removed once the last product/bundle has been removed. Useful to set to false if you want to retain existing depot ACLs for subsequent depot reuse.

rpc_binding_info=ncacn_ip_tcp:[2121] ncadg_ip_udp:[2121]
Defines the protocol sequence(s) and endpoint(s) on which the daemon listens and the other commands contact the daemon. If the connection fails for one protocol sequence, the next is attempted. SD supports both the tcp (ncacn_ip_tcp:[2121]) and udp (ncadg_ip_udp:[2121]) protocol sequence on most platforms.

See the *sd*(5) manual page (type **man 5 sd**) for more information.

rpc_timeout=5
Relative length of the communications timeout. This is a value in the range from 0 to 9 and is interpreted by the DCE RPC. Higher values mean longer times; you may need a higher value for a slow or busy network. Lower values give faster recognition on attempts to contact hosts that are not up or not running **swagentd**. Each value is approximately twice as long as the preceding value. A value of 5 is about 30 seconds for the **ncadg_ip_udp** protocol sequence. This option may not have any noticeable impact when using the **ncacn_ip_tcp** protocol sequence.

run_as_superuser=true
This option controls SD's nonprivileged mode. This option is ignored (treated as true) when the invoking user is super-user.

When set to the default value of true, SD operations are performed normally, with permissions for operations either granted to a local super-user or set by SD ACLs. (See *swacl*(1M) for details on ACLs.)

When set to false and the invoking user is local and is *not* super-user, nonprivileged mode is invoked:

- Permissions for operations are based on the user's file system permissions.
- SD ACLs are ignored.
- Files created by SD have the uid and gid of the invoking user, and the mode of created files is set according to the invoking user's umask.

SD's nonprivileged mode is intended only for managing applications that are specially designed and packaged. This mode cannot be used to manage the HP-UX operating system or patches to it. For a full explanation of nonprivileged SD, see the *Software Distributor Administration Guide*, available at the **http://docs.hp.com** web site.

See also the **admin_directory** and **installed_software_catalog** options.

software=
Defines the default *software_selections*. There is no supplied default. If there is more than one software selection, they must be separated by spaces.

software_view=products
Indicates the software view to be used by the Interactive UI of the controller. It can be set to **products**, **all_bundles**, or a bundle category tag to indicate to show only bundles of that category.

targets=
Defines the default *target_selections*. There is no supplied default (see **select_local** above). If there is more than one target selection, they must be separated by spaces.

target_shared_root=

This option applies to HP-UX 10.X only.

Defines the default location of the alternate root directory.

verbose=1
Controls the verbosity of the output (stdout). A value of
0 disables output to stdout. (Error and warning messages are always written to stderr).
1 enables verbose messaging to stdout.

write_remote_files=false
Prevents the removal of files from a remote (NFS) file system. When set to **false**, files on a remote file system are not removed.

If set to **true** and if the superuser has write permission on the remote file system, the remote files are removed.

Session File

Each invocation of **swremove** defines a task session. The command automatically saves options, source information, software selections, and target selections before the task actually commences. This lets you re-execute the command even if the session ends before the task is complete. You can also save session information from interactive or command-line sessions.

Session information is saved to the file **$HOME/.sw/sessions/swremove.last**. This file is overwritten by each invocation of the command. The file uses the same syntax as the defaults files.

From an interactive session, you can save session information into a file at any time by selecting the *Save Session* or *Save Session As* option from the *File* menu.

From a command-line session, you can save session information by executing the command with the **-C** *session__file* option. You can specify an absolute path for a session file. If you do not specify a directory, the default location is **$HOME/.sw/sessions/**.

To re-execute a saved session from an interactive session, use the *Recall Session* option from the *File* menu.

To re-execute a session from a command-line, specify the session file as the argument for the **-S** option.

When you re-execute a session file, the values in the session file take precedence over values in the system defaults file. Likewise, any command-line options and parameters take precedence over the values in the session file.

Software and Target Lists

The **swremove** command supports software and target selection from separate input files.

You can specify software and target selection lists with the **-f** and **-t** options. Software and targets specified in these files are selected for operation instead of (or in addition to) files listed in the command line. (See the **-f** and **-t** options for more information.)

Additionally, the **swremove** interactive user interface reads a default list of hosts on which to operate. The list is stored in:

/var/adm/sw/defaults.hosts	the system-wide default list of hosts
$HOME/.swdefaults.hosts	the user-specific default list of hosts

For each interactive command, target hosts containing roots or depots are specified in separate lists (**hosts** and **hosts_with_depots** respectively.) The list of hosts are enclosed in { } braces and separated by white space (blank, tab and newline). For example:

```
swremove.hosts={hostA hostB hostC hostD hostE hostF}
swremove.hosts_with_depots={hostS}
```

Environment Variables

The environment variable that affects the **swremove** command is:

LANG Determines the language in which messages are displayed. If LANG is not specified or is set to the empty string, a default value of **C** is used. See the *lang*(5) man page by typing **man 5 sd** for more information.

NOTE: The language in which the SD agent and daemon log messages are displayed is set by the system configuration variable script, **/etc/rc.config.d/LANG**. For example, **/etc/rc.config.d/LANG**, must be set to **LANG=ja_JP.SJIS** or **LANG=ja_JP.eucJP** to make the agent and daemon log messages display in Japanese.

LC_ALL Determines the locale to be used to override any values for locale categories specified by the settings of **LANG** or any environment variables beginning with **LC_**.

LC_CTYPE
Determines the interpretation of sequences of bytes of text data as characters (e.g., single- versus multibyte characters in values for vendor-defined attributes).

LC_MESSAGES
Determines the language in which messages should be written.

LC_TIME
Determines the format of dates (*create_date* and *mod_date*) when displayed by **swlist**. Used by all utilities when displaying dates and times in **stdout**, **stderr**, and **logging**.

TZ Determines the time zone for use when displaying dates and times.

Environment variables that affect scripts are:

SW_CATALOG
Holds the path to the Installed Products Database (IPD), relative to the path in the **SW_ROOT_DIRECTORY** environment variable. Note that you can specify a path for the IPD using the **installed_software_catalog** default option.

SW_CONTROL_DIRECTORY
Defines the current directory of the script being executed, either a temporary catalog directory, or a directory within in the Installed Products Database (IPD). This variable tells scripts where other control scripts for the software are located (e.g. subscripts).

SW_CONTROL_TAG
Holds the tag name of the control_file being executed. When packaging software, you can define a physical name and path for a control file in a depot. This lets you define the control_file with a name other than its tag and lets you use multiple control file definitions to point to the same file. A control_file can query the **SW_CONTROL_TAG** variable to determine which tag is being executed.

SW_LOCATION
Defines the location of the product, which may have been changed from the default product directory. When combined with the **SW_ROOT_DIRECTORY**, this variable tells scripts where the product files are located.

SW_PATH
A PATH variable which defines a minimum set of commands available for use in a control script (e.g. **/sbin:/usr/bin**).

SW_ROOT_DIRECTORY
Defines the root directory in which the session is operating, either "/" or an alternate root directory. This variable tells control scripts the root directory in which the products are installed. A script must use this directory as a prefix to **SW_LOCATION** to locate the product's installed files. The configure script is only run when **SW_ROOT_DIRECTORY** is **/**.

SW_SESSION_OPTIONS
Contains the pathname of a file containing the value of every option for a particular command, including software and target selections. This lets scripts retrieve any command options and values other than the ones provided explicitly by other environment variables. For example, when the file pointed to by **SW_SESSIONS_OPTIONS** is made available to a *request* script, the *targets* option contains a list of *software_collection_specs* for all targets specified for the command. When the file pointed to by **SW_SESSIONS_OPTIONS** is made available to other scripts, the *targets* option contains the single *software_collection_spec* for the targets on which the script is being executed.

SW_SOFTWARE_SPEC
This variable contains the fully qualified software specification of the current product or fileset. The software specification allows the product or fileset to be uniquely identified.

Additional environment variables that affect scripts for **swremove** are:

PRE_UNIX95
This variable and the **UNIX95** variable are exported with a value that forces "classic" behavior of **swremove** instead of **UNIX95** behavior. For HP-UX 10.30 and later versions, this variable is set to "1".

SW_SESSION_IS_KERNEL
Indicates whether a kernel build is scheduled for the current install/remove session. A **TRUE** value indicates that the selected kernel fileset is scheduled for a kernel build and that changes to **/stand/system** are required. A null value indicates that a kernel build is not scheduled and that changes to **/stand/system** are not required.

The value of this variable is always equal to the value of **SW_SESSION_IS_REBOOT**.

SW_SESSION_IS_REBOOT
Indicates whether a reboot is scheduled for a fileset selected for removal. Because all HP-UX kernel filesets are also reboot filesets, the value of this variables is always equal to the

value of **`SW_SESSION_IS_KERNEL`**.

`UNIX95` This variable, along with the **`PRE_U95`** variable, is exported with a value that forces "classic" behavior of **`swremove`** instead of **`UNIX95`** behavior. For the 10.30 or later release of HP-UX, this variable is cleared.

Signals

The **`swremove`** command catches the signals SIGQUIT, SIGINT, and SIGUSR1. If these signals are received, the command prints a message, sends a Remote Procedure Call (RPC) to the agents to wrap up after completion, and then exits.

The agent ignores SIGHUP, SIGINT, and SIGQUIT. It immediately exits gracefully after receiving SIGTERM, SIGUSR1, or SIGUSR2. Killing the agent may leave corrupt software on the system, and thus should only be done if absolutely necessary. Note that when an SD command is killed, the agent does not terminate until completing the task in progress.

The daemon ignores SIGHUP, SIGINT and SIGQUIT. It immediately exits gracefully after receiving SIGTERM and SIGUSR2. After receiving SIGUSR1, it waits for completion of a copy or remove from a depot session before exiting, so that it can register or unregister depots if necessary. Requests to start new sessions are refused during this wait.

Each agent will complete the removal task (if the execution phase has already started) before it wraps up. This avoids leaving software in a corrupt state.

Terminal Support

For in-depth information about terminal support refer to:

- The *Software Distributor Administration Guide*.
- Start the GUI or TUI, select the *Help* menu, then select the *Keyboard...* option to access the *Keyboard Reference Guide*.

RETURN VALUES

An interactive **`swremove`** session always returns 0. A non-interactive **`swremove`** session returns:

`0` The *software_selections* were successfully removed.
`1` The remove operation failed on all *target_selections*.
`2` The remove operation failed on some *target_selections*.

DIAGNOSTICS

The **`swremove`** command writes to stdout, stderr, and to specific log files.

Standard Output

An interactive **`swremove`** session does not write to stdout. A non-interactive **`swremove`** session writes messages for significant events. These include:

- a begin and end session message,
- selection, analysis, and execution task messages for each *target_selection*.

S

Standard Error

An interactive **`swremove`** session does not write to stderr. A non-interactive **`swremove`** session writes messages for all WARNING and ERROR conditions to stderr.

Logging

Both interactive and non-interactive **`swremove`** sessions log summary events at the host where the command was invoked. They log detailed events to the **`swagent`** logfile associated with each *target_selection*.

Command Log

A non-interactive **`swremove`** session logs all stdout and stderr messages to the the logfile **`/var/adm/sw/swremove.log`**. Similar messages are logged by an interactive **`swremove`** session. The user can specify a different logfile by modifying the **`logfile`** option.

Target Log

A **`swagent`** process performs the actual remove operation at each *target_selection*. When removing installed software, the **`swagent`** logs messages to the file **`var/adm/sw/swagent.log`** beneath the root directory (e.g. **`/`** or an alternate root directory). When removing available software (within a depot), the **`swagent`** logs messages to the file *swagent.log* beneath the depot directory (e.g. **`/var/spool/sw`**).

You can view command and target log files using the **sd** or **swjob** command.

EXAMPLES

Preview the remove of the C and Pascal products installed at the local host:

```
swremove -p cc pascal
```

Remove the C and Pascal products from several remote hosts:

```
swremove cc pascal @ hostA hostB hostC
```

Remove a particular version of HP Omniback:

```
swremove Omniback,l/opt/Omniback_v2.0
```

Remove the entire contents of a local depot:

```
swremove -d * @ /var/spool/sw
```

FILES

$HOME/.swdefaults
Contains the user-specific default values for some or all SD options. If this file does not exist, SD looks for user-specific defaults in **$HOME/.sw/defaults**.

$HOME/.sw/defaults.hosts
Contains the user-specific default list of hosts to manage.

$HOME/.sw/sessions/
Contains session files automatically saved by the SD commands, or explicitly saved by the user.

/usr/lib/sw/sys.defaults
Contains the master list of current SD options with their default values.

/var/adm/sw/
The directory which contains all of the configurable and non-configurable data for SD. This directory is also the default location of log files.

/var/adm/sw/defaults
Contains the active system-wide default values for some or all SD options.

/var/adm/sw/defaults.hosts
Contains the system-wide default list of hosts to manage.

/var/adm/sw/getdate.templ
Contains the set of date/time templates used when scheduling jobs.

/var/adm/sw/products/
The Installed Products Database (IPD), a catalog of all products installed on a system.

/var/spool/sw/
The default location of a target software depot.

AUTHOR

swremove was developed by the Hewlett-Packard Company.

SEE ALSO

swacl(1M), swagentd(1M), swask(1M), swconfig(1M), swcopy(1M), swinstall(1M), swjob(1M), swlist(1M), swmodify(1M), swpackage(1M), swreg(1M), swverify(1M), install-sd(1M), sd(4), swpackage(4), sd(5).

Software Distributor Administration Guide, available at **http://docs.hp.com**.

SD customer web site at **http://software.hp.com/SD_AT_HP/**.

NAME

swverify - verify software products

SYNOPSIS

swverify [**-d**|**-r**] [**-F**] [**-v**] [**-C** *session_file*] [**-f** *software_file*] [**-J** *job*id*]* [**-Q** *date*]
[**-S** *session_file*] [**-t** *target_file*] [**-x** *option=value*] [**-X** *option_file*]
[*software_selections*] [**@** *target_selections*]

Remarks

- This command supports operations on remote systems. See **Remote Operation** below.
- For an overview of all SD commands, see the *sd*(5) man page by typing **man 5 sd** on the command line.

DESCRIPTION

The **swverify** command verifies the *software_selections* at one or more *target_selections* (e.g. root filesystems). When verifying installed software, **swverify** checks software states, dependency relationships, file existence and integrity, in addition to executing vendor-supplied verification scripts.

The **swverify** command also verifies *software_selections* at one or more target depots. For target depots, **swverify** performs all of the checks listed above, but does not execute verification scripts.

NOTE: swverify does *not* support operations on a tape depot.

The **swverify** command also supports these features:

- Verifies whether installed or configured software is compatible with the hosts on which that software is installed.
- Verifies that all dependencies (prerequisites, corequisites, exrequisites) are being met (for installed software) or can be met (for available software).
- Executes vendor-specific **verify** scripts that check if the software products is correctly configured.
- Executes vendor-specific **fix** scripts that correct and report specific problems.
- Reports missing files, check all file attributes (ignoring volatile files). These attributes include permissions, file types, size, checksum, mtime, link source and major/minor attributes.

Remote Operation

You can enable SD to manage software on remote systems. To let the root user from a central SD *controller* (also called the *central management server* or *manager node*) perform operations on a remote *target* (also called the *host* or *agent*):

1) Install a special HP ServiceControl Manager fileset on the remote systems. This enables remote operations by automatically setting up the root, host, and template Access Control Lists (ACLs) on the remote machines and permitting root access from the controller system. To install the fileset, run the following command on each remote system:

swinstall -s *controller***:/var/opt/mx/depot11 AgentConfig.SD-CONFIG**

NOTES:

- *controller* is the name of the central management server.
- If the target is running HP-UX 10.20, use the same command but substitute **depot10** for **depot11**.
- Targets previously set up by SD/OV to be managed by this controller do not need this step.
- SD does not require any other ServiceControl Manager filesets.

2) (Optional) **swinstall**, **swcopy**, and **swremove** have enhanced GUI interfaces for remote operations. Enable the enhanced GUIs by creating the **.sdkey** file on the controller. Use this command:

touch /var/adm/sw/.sdkey

NOTES:

- This step is not required when you use SD from within the HP ServiceControl Manager.

S

- See *sd(5)*, *swinstall(1M)*, *swcopy(1M)*, *swjob(1M)*, *swlist(1M)*, or *swremove(1M)* for more information on interactive operations.

NOTE: You can also set up remote access by using **swacl** directly on the remote machines to grant root or non-root access to users from the controller system.

Options

swverify supports the following options:

-d Operate on a depot rather than installed software.

-F Runs vendor-specific **fix** scripts to correct and report problems on installed software. The fix script can create missing directories, correct file modifications (mode, owner, group, major, and minor), and recreate symbolic links.

-r Operates on an alternate root directory, which must be specified the **@** *target_selections* option. Verify scripts are not run when verifying software in an alternate root directory. (This option is not required for alternate root operations but is maintained for backward compatibility. See the **Alternate Root Directory and Depot Directory** heading in *sd*(5) for more information.)

-v Turns on verbose output to stdout. (The **swverify** logfile is not affected by this option.) Verbose output is enabled by default; see the **verbose** option below.

-C *session_file*
Save the current options and operands to *session_file*. You can enter a relative or absolute path with the file name. The default directory for session files is **$HOME/.sw/sessions/**. You can recall a session file with the **-S** option.

-f *software_file*
Read the list of *software_selections* from *software_file* instead of (or in addition to) the command line.

-J *jobid* (HP OpenView Software Distributor only) Executes the previously scheduled job. This is the syntax used by the daemon to start the job.

-Q *date* (HP OpenView Software Distributor only) Schedules the job for this date. You can change the date format by editing the **/var/adm/sw/getdate.templ** file.

-S *session_file*
Execute **swverify** based on the options and operands saved from a previous session, as defined in *session_file*. You can save session information to a file with the **-C** option.

-t *target_file* Read the list of *target_selections* from *target_file* instead of (or in addition to) the command line.

-x *option=value*
Set the session *option* to *value* and override the default value (or a value in an alternate *options_file* specified with the **-X** option). Multiple **-x** options can be specified.

-X *option_file* Read the session options and behaviors from *options_file*.

Operands

Most SD commands support two types of operands: *software selections* followed by *target selections*. These operands are separated by the **@** (at) character. This syntax implies that the command operates on "software selections at targets".

Software Selections

The **swverify** command supports the following syntax for each *software_selection*:

bundle[**.***product*[**.***subproduct*][**.***fileset*]][**,***version*]

product[**.***subproduct*][**.***fileset*][**,***version*]

- The **=** (equals) relational operator lets you specify selections with the following shell wildcard and pattern-matching notations:

  [ ], *****, **?**

S

- *Bundles* and *subproducts* are recursive. *Bundles* can contain other *bundles* and *subproducts* can contain other *subproducts*.
- The ***** software specification selects all products. Use this specification with caution.

The **version** component has the form:

[**,r** *<op> revision*][**,a** *<op> arch*][**,v** *<op> vendor*]
[**,c** *<op> category*][**,q=***qualifier*][**,l=***location*]
[**,fr** *<op> revision*][**,fa** *<op> arch*]

- *location* applies only to installed software and refers to software installed to a location other than the default product directory.
- **fr** and **fa** apply only to filesets.
- The *<op>* (relational operator) component can be of the form:

  =, **==**, **>=**, **<=**, **<**, **>**, or **!=**

  which performs individual comparisons on dot-separated fields.

  For example, **r>=B.10.00** chooses all revisions greater than or equal to **B.10.00**. The system compares each dot-separated field to find matches.
- The **=** (equals) relational operator lets you specify selections with the shell wildcard and pattern-matching notations:

  [ ], *****, **?**, **!**

  For example, the expression **r=1[01].*** returns any revision in version 10 or version 11.
- All version components are repeatable within a single specification (e.g. **r>=A.12**, **r<A.20**). If multiple components are used, the selection must match all components.
- Fully qualified software specs include the **r=**, **a=**, and **v=** version components even if they contain empty strings. For installed software, **l=** is also included.
- No space or tab characters are allowed in a software selection.
- The software *instance_id* can take the place of the version component. It has the form:

  [*instance_id*]

  within the context of an exported catalog, where *instance_id* is an integer that distinguishes versions of products and bundles with the same tag.

Target Selections

The **swverify** command supports the following syntax for each *target_selection*. The : (colon) is required if both a host and directory are specified.

[*host*][**:**][**/***directory*]

EXTERNAL INFLUENCES

Default Options

In addition to the standard options, several SD behaviors and policy options can be changed by editing the default values found in:

/var/adm/sw/defaults	the system-wide default values.
$HOME/.swdefaults	the user-specific default values.

Values must be specified in the defaults file using this syntax:

[*command_name***.**]*option***=***value*

The optional *command_name* prefix denotes one of the SD commands. Using the prefix limits the change in the default value to that command. If you leave the prefix off, the change applies to all commands.

You can also override default values from the command line with the **-x** or **-X** options:

command **-x** *option***=***value*

command **-X** *option_file*

The following section lists all of the keywords supported by the **swverify** command. If a default value exists, it is listed after the "=". The commands that this option applies to are also specified.

admin_directory=/var/adm/sw (for normal mode)
admin_directory=/var/home/LOGNAME/sw (for nonprivileged mode)

The location for SD logfiles and the default parent directory for the installed software catalog. The default value is **/var/adm/sw** for normal SD operations. When SD operates in nonprivileged mode (that is, when the **run_as_superuser** default option is set to **true**):

- The default value is forced to **/var/home/LOGNAME/sw**.
- The path element **LOGNAME** is replaced with the name of the invoking user, which SD reads from the system password file.
- If you set the value of this option to **HOME/***path*, SD replaces **HOME** with the invoking user's home directory (from the system password file) and resolves *path* relative to that directory. For example, **HOME/my_admin** resolves to the **my_admin** directory in your home directory.
- If you set the value of the **installed_software_catalog** default option to a relative path, that path is resolved relative to the value of this option.

SD's nonprivileged mode is intended only for managing applications that are specially designed and packaged. This mode cannot be used to manage the HP-UX operating system or patches to it. For a full explanation of nonprivileged SD, see the *Software Distributor Administration Guide*, available at the **http://docs.hp.com** web site.

See also the **installed_software_catalog** and **run_as_superuser** options.

agent_auto_exit=true

Causes the target agent to automatically exit after Execute phase, or after a failed Analysis phase. This is forced to **false** when the controller is using an interactive UI, or when **-p** (preview) is used. This enhances network reliability and performance. The default value of **true** causes the target agent to automatically exit when appropriate. When set to **false**, the target agent will not exit until the controller ends the session.

agent_timeout_minutes=10000

Causes a target agent to exit if it has been inactive for the specified time. This can be used to make target agents more quickly detect lost network connections since RPC can take as long as 130 minutes to detect a lost connection. The recommended value is the longest period of inactivity expected in your environment. For command line invocation, a value between 10 minutes and 60 minutes is suitable. A value of 60 minutes or more is recommended when the GUI will be used. The default of 10000 is slightly less than 7 days.

allow_incompatible=false

Requires that the software products which are being installed be "compatible" with the target selections. (All of the target selections must match the list of supported systems defined for each selected product.) If set to **true**, target compatibility is not enforced.

allow_multiple_versions=false

Prevents the installation or configuration of another, independent version of a product when a version already is installed or configured at the target.

If set to **true**, another version of an existing product can be installed into a new location, or can be configured in its new location. Multiple versions can only be installed if a product is locatable. Multiple configured versions will not work unless the product supports it.

autoremove_job=false

This option applies only to HP OpenView Software Distributor.

Controls automatic job removal of completed jobs. If the job is automatically removed, job information (job status or target log files) cannot be queried with **swjob**.

autoselect_dependencies=true

Controls the automatic selection of prerequisite, corequisite, and exrequisite software that is not explicitly selected by the user. When set to **true**, the requisite software is automatically selected for configuration. When set to **false**, requisite software which is not explicitly selected is not automatically selected for configuration.

check_contents=true
Causes **swverify** to verify the time stamp, size, and checksum attributes of files. If set to **false**, these attributes are not verified.

check_contents_uncompressed=false
(This option is ignored if **check_contents** is set to **false**.) Controls whether or not **swverify** validates the size and checksum for compressed files. In the default state of **false**, **swverify** checks only the mtime, size and cksum attributes of the compressed file. If set to **true**, **swverify** uncompresses the file in memory and verifies the size and cksum attributes of the uncompressed contents.

Only files compressed with SD's internal compressor can be uncompressed during a **swverify** operation. See the **compress_files** option of the *swpackage*(1M) command for more information.

check_contents_use_cksum=true
(This option is ignored if **check_contents** is set to **false**.) Controls whether or not **swverify** computes a checksum on the contents of the file. In the default state of **true**, **swverify** checks all file attributes including the checksum. If set to **false**, **swverify** checks only the file timestamp and size.

check_permissions=true
Causes **swverify** to verify the mode, owner, UID, group, and GID attributes of installed files. If set to **false**, these attributes are not verified.

check_requisites=true
Causes **swverify** to verify that the prerequisite, corequisite, and exrequisite dependencies of the software selections are being met. If set to **false**, these checks are not performed.

check_scripts=true
Causes **swverify** to run the fileset/product verify scripts for installed software. If set to **false**, these scripts are not executed.

check_volatile=false
Causes **swverify** to not verify those files marked as volatile (i.e. can be changed). If set to **true**, volatile files are also checked (for installed software).

controller_source=
Specifies the location of a depot for the controller to access to resolve selections. Setting this option can reduce network traffic between the controller and the target. Use the target selection syntax to specify the location:

[*host*][**:**][*path*]

This option has no effect on which sources the target uses.

S

distribution_target_directory=/var/spool/sw
Defines the default distribution directory of the target depot. The *target_selection* operand overrides this default.

enforce_dependencies=true
Requires that all dependencies specified by the *software_selections* be resolved either in the specified source, or at the *target_selections* themselves.

If set to **false**, dependencies will still be checked, but not enforced. Corequisite dependencies, if not enforced, may keep the selected software from working properly. Prerequisite or exrequisite dependencies, if not enforced, may cause the installation or configuration to fail.

enforce_locatable=true
(Currently, **swverify** recognizes this option, but the option has no associated behavior. See *swinstall*(1M) or *sd*(5) for more information.) Controls the handling of errors when relocating a non-relocatable fileset. If **true**, an error is generated when an attempt is made to relocate a non-relocatable fileset. If **false**, an attempt is made to relocate the fileset in any case.

fix=false
If **true**, runs vendor-specific scripts to correct and report problems on installed software. Fix scripts can create missing directories, correct file modifications, (mode, owner, group,

major, minor), and recreate symbolic links. If **false**, fix scripts are not run.

installed_software_catalog=products

Defines the directory path where the Installed Products Database (IPD) is stored. This information describes installed software. When set to an absolute path, this option defines the location of the IPD. When this option contains a relative path, the SD controller appends the value to the value specified by the **admin_directory** option to determine the path to the IPD. For alternate roots, this path is resolved relative to the location of the alternate root. This option does not affect where software is installed, only the IPD location.

This option permits the simultaneous installation and removal of multiple software applications by multiple users or multiple processes, with each application or group of applications using a different IPD.

Caution: use a specific **installed_software_catalog** to manage a specific application. SD does not support multiple descriptions of the same application in multiple IPDs.

See also the **admin_directory** and **run_as_superuser** options, which control SD's nonprivileged mode. (This mode is intended only for managing applications that are specially designed and packaged. This mode cannot be used to manage the HP-UX operating system or patches to it. For a full explanation of nonprivileged SD, see the *Software Distributor Administration Guide*, available at the **http://docs.hp.com** web site.)

job_title=

This option applies only to HP OpenView Software Distributor.

This is an ASCII string giving a title to a job. It is displayed along with the job ID to provide additional identifying information about a job when **swjob** is invoked.

logdetail=false[true]

Controls the amount of detail written to the logfile. When set to **true**, this option adds detailed task information (such as options specified, progress statements, and additional summary information) to the logfile. This information is in addition to log information controlled by the **loglevel** option.

logfile=/var/adm/sw/sw<command>.log

Defines the default log file for each SD command. (The agent log files are always located relative to the target depot or target root, e.g. **/var/spool/sw/swagent.log** and **/var/adm/sw/swagent.log**.)

loglevel=1

Controls the log level for the events logged to the command logfile, the target agent logfile, and the source agent logfile. This information is in addition to the detail controlled by the **logdetail** option. See **logdetail**, above, and the *sd*(5) manual page (by typing **man 5 sd**) for more information. A value of

0 provides no information to the logfile.
1 enables verbose logging to the logfiles.
2 enables very verbose logging to the logfiles.

log_msgid=0

Adds numeric identification numbers at the beginning of SD logfile messages:

0 (default) No identifiers are attached to messages.
1 Adds identifiers to ERROR messages only.
2 Adds identifiers to ERROR and WARNING messages.
3 Adds identifiers to ERROR, WARNING, and NOTE messages.
4 Adds identifiers to ERROR, WARNING, NOTE, and certain other informational messages.

mount_all_filesystems=true

By default, the SD commands attempt to mount all filesystems in the */etc/fstab* file at the beginning of the analysis phase, to ensure that all listed filesystems are mounted before proceeding. This policy helps to ensure that files are not loaded into a directory that may be below a future mount point, and that the expected files are available for a remove or verify operation.

If set to **false**, the mount operation is not attempted, and no check of the current mounts is performed.

`rpc_binding_info=ncacn_ip_tcp:[2121] ncadg_ip_udp:[2121]`
Defines the protocol sequence(s) and endpoint(s) on which the daemon listens and which the other commands use to contact the daemon. If the connection fails for one protocol sequence, the next is attempted. SD supports both the tcp (ncacn_ip_tcp:[2121]) and udp (ncadg_ip_udp:[2121]) protocol sequence on most platforms. See the *sd*(5) man page by typing **`man 5 sd`** for more information.

`rpc_timeout=5`
Relative length of the communications timeout. This is a value in the range from 0 to 9 and is interpreted by the DCE RPC. Higher values mean longer times; you may need a higher value for a slow or busy network. Lower values will give faster recognition on attempts to contact hosts that are not up, or are not running **`swagentd`**. Each value is approximately twice as long as the preceding value. A value of 5 is about 30 seconds for the **`ncadg_ip_udp`** protocol sequence. This option may not have any noticeable impact when using the **`ncacn_ip_tcp`** protocol sequence.

`run_as_superuser=true`
This option controls SD's nonprivileged mode. This option is ignored (treated as true) when the invoking user is super-user.

When set to the default value of true, SD operations are performed normally, with permissions for operations either granted to a local super-user or set by SD ACLs. (See *swacl*(1M) for details on ACLs.)

When set to false and the invoking user is local and is *not* super-user, nonprivileged mode is invoked:

- Permissions for operations are based on the user's file system permissions.
- SD ACLs are ignored.
- Files created by SD have the uid and gid of the invoking user, and the mode of created files is set according to the invoking user's umask.

SD's nonprivileged mode is intended only for managing applications that are specially designed and packaged. This mode cannot be used to manage the HP-UX operating system or patches to it. For a full explanation of nonprivileged SD, see the *Software Distributor Administration Guide*, available at the **`http://docs.hp.com`** web site.

See also the **`admin_directory`** and **`installed_software_catalog`** options.

`select_local=true`
If no *target_selections* are specified, select the default **`target_directory`** of the local host as the *target_selection* for the command.

`software=`
Defines the default *software_selections*. There is no supplied default. If there is more than one software selection, they must be separated by spaces. Software is usually specified in a software input file, as operands on the command line, or in the GUI.

S

`targets=`
Defines the default *target_selections*. There is no supplied default (see **`select_local`** above). If there is more than one target selection, they must be separated by spaces. Targets can be specified in a target input file or as operands on the command line.

`verbose=1`
Controls the verbosity of a non-interactive command's output:

`0`	disables output to stdout. (Error and warning messages are always written to stderr).
`1`	enables verbose messaging to stdout.
`2`	for **`swpackage`** and **`swmodify`**, enables very verbose messaging to stdout.

The **`-v`** option overrides this default if it is set to 0.

Session File

Each invocation of the **`swverify`** command defines a verify session. The invocation options, source information, software selections, and target hosts are saved before the installation or copy task actually commences. This lets you re-execute the command even if the session ends before proper completion.

Each session is saved to the file **`$HOME/.sw/sessions/swverify.last`**. This file is overwritten by each invocation of **`swverify`**.

You can also save session information to a specific file by executing **swverify** with the **-C** *session__file* option.

A session file uses the same syntax as the defaults files. You can specify an absolute path for the session file. If you do not specify a directory, the default location for a session file is **$HOME/.sw/sessions/**.

To re-execute a session file, specify the session file as the argument for the **-S** *session__file* option of **swverify**.

Note that when you re-execute a session file, the values in the session file take precedence over values in the system defaults file. Likewise, any command line options or parameters that you specify when you invoke **swverify** take precedence over the values in the session file.

Environment Variables

SD programs that execute control scripts set environment variables for use by the control scripts.

The environment variable that affects the **swverify** command is:

LANG Determines the language in which messages are displayed. If LANG is not specified or is set to the empty string, a default value of **C** is used. See the *lang*(5) man page by typing **man 5 sd** for more information.

NOTE: The language in which the SD agent and daemon log messages are displayed is set by the system configuration variable script, **/etc/rc.config.d/LANG**. For example, **/etc/rc.config.d/LANG**, must be set to **LANG=ja_JP.SJIS** or **LANG=ja_JP.eucJP** to make the agent and daemon log messages display in Japanese.

LC_ALL Determines the locale to be used to override any values for locale categories specified by the settings of **LANG** or any environment variables beginning with **LC_**.

LC_CTYPE
Determines the interpretation of sequences of bytes of text data as characters (e.g., single- versus multibyte characters in values for vendor-defined attributes).

LC_MESSAGES
Determines the language in which messages should be written.

LC_TIME
Determines the format of dates (*create_date* and *mod_date*) when displayed by **swlist**. Used by all utilities when displaying dates and times in **stdout**, **stderr**, and **logging**.

TZ Determines the time zone for use when displaying dates and times.

Environment variables that affect scripts:

SW_CATALOG
Holds the path to the Installed Products Database (IPD), relative to the path in the **SW_ROOT_DIRECTORY** environment variable. Note that you can specify a path for the IPD using the **installed_software_catalog** default option.

SW_CONTROL_DIRECTORY
Defines the current directory of the script being executed, either a temporary catalog directory, or a directory within in the Installed Products Database (IPD). This variable tells scripts where other control scripts for the software are located (e.g. subscripts).

SW_CONTROL_TAG
Holds the tag name of the control_file being executed. When packaging software, you can define a physical name and path for a control file in a depot. This lets you define the control_file with a name other than its tag and lets you use multiple control file definitions to point to the same file. A control_file can query the **SW_CONTROL_TAG** variable to determine which tag is being executed.

SW_LOCATION
Defines the location of the product, which may have been changed from the default product directory. When combined with the **SW_ROOT_DIRECTORY**, this variable tells scripts where the product files are located.

SW_PATH
A PATH variable which defines a minimum set of commands available to for use in a control script (e.g. **/sbin:/usr/bin**).

SW_ROOT_DIRECTORY
Defines the root directory in which the session is operating, either **/** or an alternate root directory. This variable tells control scripts the root directory in which the products are installed. A script must use this directory as a prefix to **SW_LOCATION** to locate the product's installed files. The configure script is only run when **SW_ROOT_DIRECTORY** is **/**.

SW_SESSION_OPTIONS
Contains the pathname of a file containing the value of every option for a particular command, including software and target selections. This lets scripts retrieve any command options and values other than the ones provided explicitly by other environment variables. For example, when the file pointed to by **SW_SESSIONS_OPTIONS** is made available to a *request* script, the *targets* option contains a list of *software_collection_specs* for all targets specified for the command. When the file pointed to by **SW_SESSIONS_OPTIONS** is made available to other scripts, the *targets* option contains the single *software_collection_spec* for the targets on which the script is being executed.

SW_SOFTWARE_SPEC
This variable contains the fully qualified software specification of the current product or fileset. The software specification allows the product or fileset to be uniquely identified.

Signals

The **swverify** command catches the signals SIGQUIT, SIGINT, and SIGUSR1. If these signals are received, the command prints a message, sends a Remote Procedure Call (RPC) to the agents to wrap up after completion, and then exits.

The agent ignores SIGHUP, SIGINT, and SIGQUIT. It immediately exits gracefully after receiving SIGTERM, SIGUSR1, or SIGUSR2. Killing the agent may leave corrupt software on the system, and thus should only be done if absolutely necessary. Note that when an SD command is killed, the agent does not terminate until completing the task in progress.

The daemon ignores SIGHUP, SIGINT and SIGQUIT. It immediately exits gracefully after receiving SIGTERM and SIGUSR2. After receiving SIGUSR1, it waits for completion of a copy or remove from a depot session before exiting, so that it can register or unregister depots if necessary. Requests to start new sessions are refused during this wait.

RETURN VALUES

The **swverify** command returns:

0 The *software_selections* were successfully verified.
1 The verify operation failed on all *target_selections*.
2 The verify operation failed on some *target_selections*.

DIAGNOSTICS

The **swverify** command writes to stdout, stderr, and to specific logfiles.

S

Standard Output

The **swverify** command writes messages for significant events. These include:

- a begin and end session message,
- selection, analysis, and execution task messages for each *target_selection*.

Standard Error

The **swverify** command also writes messages for all WARNING and ERROR conditions to stderr.

Logging

The **swverify** command logs summary events at the host where the command was invoked. It logs detailed events to the **swagent** logfile associated with each *target_selection*.

Command Log
The **swverify** command logs all stdout and stderr messages to the the logfile **/var/adm/sw/swverify.log**. (The user can specify a different logfile by modifying the **log-file** option.)

Target Log
A **swagent** process performs the actual verify operation at each *target_selection*. When verifying installed software, the **swagent** logs messages to the file **var/adm/sw/swagent.log** beneath the root directory (e.g. **/** or an alternate root directory). When verifying available software (within

a depot), the **swagent** logs messages to the file *swagent.log* beneath the depot directory (e.g. **/var/spool/sw**).

The following line applies only to HP OpenView Software Distributor.

Command and target log files can be viewed using the **swjob** command.

EXAMPLES

Verify the C and Pascal products installed at the local host:

```
swverify cc pascal
```

Verify a particular version of HP Omniback:

```
swverify Omniback,1=/opt/Omniback_v2.0
```

Verify the entire contents of a local depot:

```
swverify -d \* @ /var/spool/sw
```

The following example applies only to HP OpenView Software Distributor.

Verify the C and Pascal products on remote hosts:

```
swverify cc pascal @ hostA hostB hostC
```

FILES

$HOME/.swdefaults
Contains the user-specific default values for some or all SD options.

$HOME/.sw/sessions/
Contains session files automatically saved by the SD commands, or explicitly saved by the user.

/usr/lib/sw/sys.defaults
Contains the master list of current SD options with their default values.

/var/adm/sw/
The directory which contains all the configurable and non-configurable data for SD. This directory is also the default location of logfiles.

/var/adm/sw/defaults
Contains the active system-wide default values for some or all SD options.

/var/adm/sw/getdate.templ
Contains the set of date/time templates used when scheduling jobs.

/var/adm/sw/products/
The Installed Products Database (IPD), a catalog of all products installed on a system.

/var/spool/sw/
The default location of a target software depot.

AUTHOR

swverify was developed by the Hewlett-Packard Company.

SEE ALSO

install-sd(1M), swacl(1M), swagentd(1M), swask(1M), swconfig(1M), swcopy(1M), swinstall(1M), swjob(1M), swlist(1M), swmodify(1M), swpackage(1M), swreg(1M), swremove(1M), sd(4), swpackage(4), sd(5).

Software Distributor Administration Guide, available at **http://docs.hp.com**.

SD customer web site at **http://software.hp.com/SD_AT_HP/**.

NAME

sync - synchronize file systems

SYNOPSIS

sync [**-l**]

DESCRIPTION

sync executes the **sync()** system call (see *sync*(2)). If the system is to be stopped, the **sync** command must be called to ensure file system integrity.

sync flushes all previously unwritten system buffers including modified super blocks, modified inodes, and delayed block I/O out to disk. This ensures that all file modifications are properly saved before performing a critical operation such as a system shutdown. For additional protection from power failures or possible system crashes, use **syncer** to execute **sync** automatically at periodic intervals (see *syncer*(1M)).

AUTHOR

sync was developed by AT&T and HP.

SEE ALSO

syncer(1M), sync(2).

STANDARDS CONFORMANCE

sync: SVID2, SVID3

NAME

syncer - periodically sync for file system integrity

SYNOPSIS

/usr/sbin/syncer [*seconds*] [**-s**] [**-d** *directory* ...]

DESCRIPTION

syncer is a program that periodically executes **sync()** at an interval determined by the input argument *seconds* (see *sync*(2)). If *seconds* is not specified, the default interval is every 30 seconds. This ensures that the file system is fairly up-to-date in case of a crash. This command should not be executed directly, but should be executed at system boot time via startup script **/sbin/init.d/syncer**.

syncer also updates the **/etc/mnttab** file if it does not match current kernel mount information.

Options

syncer recognizes the following options:

-s Cause **syncer** to not update the **/etc/mnttab** file. Use of this option is provided for special cases of backward compatilibity only, and is strongly discouraged. This option may be removed in a future release.

-d Open directories for cache benefit. All directories must be specified by their full path name. If the **-d** option is not used, no directories are opened.

AUTHOR

syncer was developed by the University of California, Berkeley and HP.

SEE ALSO

init(1M), sync(1M), sync(2).

NAME

sysdef - display system definition

SYNOPSIS

/usr/sbin/sysdef [*kernel* [*master*]]

DESCRIPTION

The command **sysdef** analyzes the currently running system and reports on its tunable configuration parameters. *kernel* and *master* are not used, but can be specified for standards compliance.

For each configuration parameter, the following information is printed:

NAME	The name and description of the parameter.
VALUE	The current value of the parameter.
BOOT	The value of the parameter at boot time, if different from the current value.
MIN	The minimum allowed value of the parameter, if any.
MAX	The maximum allowed value of the parameter, if any.
UNITS	Where appropriate, the units by which the parameter is measured.
FLAGS	Flags that further describe the parameter. The following flag is defined:

M Parameter can be modified without rebooting.

WARNINGS

Users of **sysdef** must not rely on the exact field widths and spacing of its output, as these will vary depending on the system, the release of HP-UX, and the data to be displayed.

FILES

/usr/conf/master.d Directory containing master files

NAME

syslogd - log system messages

SYNOPSIS

/usr/sbin/syslogd [**-a**] [**-d**] [**-D**] [**-f** *configfile*] [**-m** *markinterval*] [**-N**] [**-p** *logfile*] [**-r**]

DESCRIPTION

The **syslogd** command reads and logs messages into a set of files described by the configuration file **/etc/syslog.conf**.

Options

syslogd recognizes the following options:

-a	Allows all messages except consecutive duplicate messages without reordering them.
-d	Turn on debugging.
-D	Prevent the kernel from directly printing its messages on the system console. In this case, **syslogd** is responsible for routing all kernel messages to their proper destination.
-f *configfile*	Use *configfile* instead of **/etc/syslog.conf**.
-m *markinterval*	Wait *markinterval* minutes between mark messages, instead of 20 minutes.
-N	Don't listen to socket.
-p *logfile*	Use *logfile* instead of **/dev/log**.
-r	Don't suppress duplicate messages.

syslogd creates the file **/var/run/syslog.pid**, if possible, containing a single line with its process ID. This can be used to kill or reconfigure **syslogd**.

To kill **syslogd**, send it a terminate signal:

```
kill ‘cat /var/run/syslog.pid‘
```

To make **syslogd**, re-read its configuration file, send it a **HANGUP** signal:

```
kill -HUP ‘cat /var/run/syslog.pid‘
```

syslogd collects messages from the UNIX domain socket **/dev/log.un**, an Internet domain socket specified in **/etc/services**, the named pipe **/dev/log**, and from the kernel log device **/dev/klog**. By default, local programs calling **syslog()** send log messages to the UNIX domain socket (see *syslog*(3C)). If UNIX domain sockets are not configured on the system, they write to the named pipe instead. If INET domain sockets are not configured, **syslogd** does not receive messages forwarded from other hosts, nor does it forward messages (see below).

Each message is one line. A message can contain a priority code and facility code as the second field of the line. By default, the priority and facility codes are not logged in log files. To add the priority and facility codes in the message field, the user has to add **-v** in the **/etc/rc.config.d/syslogd** file and restart the daemon. Priorities and Facilities are defined in the header file **<syslog.h>**.

syslogd configures itself when it starts up and whenever it receives a hangup signal. Lines in the configuration file consist of a **selector** to determine the message priorities to which the line applies and an **action**. The *action* field is separated from the selector by one or more tabs.

Selectors are semicolon separated lists of priority specifiers. Each priority has a **facility** indicating the subsystem that generated the message, a dot, and a **level** indicating the severity of the message. Symbolic names can be used. An asterisk selects all facilities. All messages of the specified level or higher (greater severity) are selected. More than one facility can be selected, using commas to separate them. For example:

```
*.emerg;mail,daemon.crit
```

selects all facilities at the **emerg** level and the **mail** and **daemon** facilities at the **crit** level.

The known facilities and levels recognized by **syslogd** are those listed in *syslog*(3C) converted to lowercase without the leading **LOG_**. The additional facility **mark** has a message at priority **LOG_INFO** sent to it every 20 minutes (this can be changed with the **-m** flag). The **mark** facility is not enabled by a facility

S

field containing an asterisk. The level **none** can be used to disable a particular facility. For example,

```
*.debug;mail.none
```

selects all messages except **mail** messages.

The second part of each line describes where the message is to be logged if this line is selected. There are four forms:

- A file name (beginning with a leading slash). The file is opened in append mode. If the file does not exist, it is created.
- A host name preceded by an **@** character. Selected messages are forwarded to the **syslogd** on the named host.
- A comma-separated list of users. Selected messages are written to those users' terminals if they are logged in.
- An asterisk. Selected messages are written to the terminals of all logged-in users.

Blank lines and lines beginning with a **#** character are ignored.

For example, the configuration file:

```
kern,mark.debug   /dev/console
mail.debug        /var/adm/syslog/mail.log
*.info;mail.none  /var/adm/syslog/syslog.log
*.alert           /dev/console
*.alert           root,eric,kridle
*.emerg           *
*.emerg           @admin
```

logs all kernel messages and 20 minute marks onto the system console, all mail system messages to **/var/adm/syslog/mail.log**, and all messages at **info** and above, except mail messages, to the file **/var/adm/syslog/syslog.log**. Messages at **alert** and above are logged to the console and to the users **root**, **eric**, and **kridle** if they are logged in. **emerg** messages are written to all logged-in users' terminals, and forwarded to the host **admin**.

Only a superuser can invoke **syslogd**.

WARNINGS

A configuration file selector selects all messages at the specified level *or higher*. The configuration lines:

```
user.debug         /tmp/logfile
user.info          /tmp/logfile
```

cause the logfile to get *two* copies of all **user** messages at level **info** and above.

Kernel panic messages are not sent to **syslogd.**

All HP-UX kernel messages are treated as if they had the **crit** priority level.

If **syslogd** is invoked with the **-D** option and **syslogd** terminates abnormally, kernel messages will not appear on the system console. In that case, reinvoke **syslogd** without the **-D** option to enable the kernel to send its messages to the system console.

DEPENDENCIES

Series 700

Kernel logging through the special log device **/dev/klog** is not supported.

The **-D** option is not supported.

AUTHOR

syslogd was developed by the University of California, Berkeley.

FILES

/dev/klog	The kernel log device
/dev/log	The named pipe on which **syslogd** reads log messages
/dev/log.un	The UNIX domain socket on which **syslogd** reads log messages
/etc/syslog.conf	Configuration file

`/var/run/syslog.pid` Process ID

SEE ALSO

logger(1), syslog(3C).

NAME

talkd - remote user communication server

SYNOPSIS

talkd

DESCRIPTION

Talkd is the server that notifies a user that someone wants to initiate a conversation. It acts as a repository of invitations, responding to requests by clients wishing to initiate a conversation. To initiate a conversation, the client (the **talk** command) sends a message of type LOOK_UP to the server (see **/usr/include/protocols/talkd.h**). This causes the server to search its invitation table to check if an invitation currently exists for the caller to talk to the callee specified in the message. If the lookup fails, the caller then sends a message of type ANNOUNCE, which causes the server to display an announcement on the callee's terminal requesting contact. When the callee responds, the local server uses the recorded invitation to respond with the appropriate rendezvous address and the caller and callee client programs establish a stream connection through which the conversation takes place.

To activate the talk service, the following entry must be added to the **/etc/inetd.conf** file:

```
ntalk  dgram  udp  wait  root  /usr/lbin/ntalkd ntalkd
```

SEE ALSO

talk(1), write(1).

NAME

telnetd - TELNET protocol server

SYNOPSIS

/usr/lbin/telnetd [**-b** [*bannerfile*]] [**-s**] [**-t**] [**-z**] [**-TCP_DELAY**]

DESCRIPTION

The **telnetd** daemon executes a server that supports the DARPA standard TELNET virtual terminal protocol. The Internet daemon (**inetd**) executes **telnetd** when it receives a service request at the port listed in the services data base for **telnet** using the **tcp** protocol (see *inetd*(1M) and *services*(4)).

telnetd operates by allocating a Telnet pseudo-terminal device (see *tels*(7)) for a client, then creating a login process which has the slave side of the Telnet pseudo-terminal as **stdin**, **stdout**, and **stderr**. **telnetd** manipulates the master side of the Telnet pseudo-terminal, implementing the TELNET protocol, and passing characters between the client and login process.

NOTE: **telnetd** no longer uses *pty*(7) devices; instead it uses special devices created for TELNET sessions only. For a full description, see *tels*(7).

When a TELNET session is started up, **telnetd** sends TELNET options to the client side, indicating a willingness to do *remote echo* of characters, to *suppress go ahead*, and to receive *terminal speed* and *terminal type* information from the remote client. If the remote client is willing, the remote terminal type is propagated in the environment of the created login process. The pseudo-terminal allocated to the client is configured as a normal terminal for login, with the exception of echoing characters (see *tty*(7)).

> **telnetd** is willing to *do*: *echo*, *binary*, *suppress go ahead*, and *timing mark*.
>
> **telnetd** is willing to have the remote client *do*: *binary*, *flow control*, *terminal speed*, *terminal type*, and *suppress go ahead*.

The flow control option permits applications running on a remote host to toggle the flow control on the local host. To toggle flow control for a **telnet** session programmatically, the application program must first call the **tcgetattr** function to get the current **termios** settings. For example,

```
tcgetattr(filedes, &termios_p)
```

Then, the **c_iflag** of the **termios** structure must have **IXON** set(reset) to enable(disable) flow control.

Finally, the **tcsetattr** function call can implement the change. For example,

```
tcsetattr(filedes, TCSANOW, &termios_p)
```

To toggle the flow control interactively, the user can issue a **stty** command using the input options **-ixon** to disable, or **ixon** to enable flow control. (see *stty*(1)).

The terminal speed option permits applications running on a remote host to obtain the terminal speed of the local host session using either *ioctl* or *stty*.

The **telnet** server also supports the TAC User ID (also known as the TAC Access Control System, or TACACS User ID) option, whereby users telneting to two or more consenting hosts may avoid going through a second login sequence. See the **-t** option below.

To start **telnetd** from the Internet daemon, the configuration file **/etc/inetd.conf** must contain an entry as follows:

```
telnet stream tcp nowait root /usr/lbin/telnetd telnetd
```

telnet uses the same files as **rlogin** to verify participating systems and authorized users, **hosts.equiv** and **.rhosts**. (See *hosts.equiv*(4) and the *Managing Systems and Workgroups* manual for configuration details.)

Options

telnetd has the following options.

-b [*bannerfile*] Specify a file containing a custom banner. This option overrides the standard **telnetd** login banner. For example, to use **/etc/issue** as the login banner, have **inetd** start **telnetd** with the following lines in **/etc/inetd.conf** (\ provides line continuation):

```
telnet stream tcp nowait root /usr/lbin/telnetd \
     telnetd -b/etc/issue
```

If *bannerfile* is not specified, **telnetd** does not print a login banner.

-s This options allows users to set the BUFFERSIZE value. This options, when set, informs **telnetd** the number of user bytes to concatenate before sending to TCP. This option is set with integer values. There is no specified default.

-t Enable the TAC User ID option. The system administrator can enable the TAC User ID option on servers designated as participating hosts by having **inetd** start **telnetd** with the **-t** option in **/etc/inetd.conf**:

```
telnet stream tcp nowait root /usr/lbin/telnetd telnetd -t
```

In order for the TAC User ID option to work as specified, the system administrator must assign to all authorized users of the option the same login name and unique user ID (UUID) on every participating system to which they are allowed TAC User ID access. These same UUIDs should not be assigned to non-authorized users.

Users cannot use the feature on systems where their local and remote UUIDs differ, but they can always use the normal **telnet** login sequence. Also, there may be a potential security breach where a user with one UUID may be able to gain entry to participating systems and accounts where that UUID is assigned to someone else, unless the above restrictions are followed.

A typical configuration may consist of one or more secure front-end systems and a network of participating hosts. Users who have successfully logged onto the front-end system may **telnet** directly to any participating system without being prompted for another login.

-z This option allows users to set the BUFFERTIMEOUT value. This option, when set, informs **telnetd** how long it should wait before timing out and flushing the concatenated user data to TCP. Note that the TIMEOUT value is measured in clock ticks (10ms) and not in seconds. This option is set with integer values. There is no specified default.

-TCP_DELAY This option allows the users to disable the **TCP_NODELAY** socket option. When **telnetd** is invoked with this option, small writes over **telnetd** may concatenate at the tcp level so that larger tcp packets are sent to the client at less frequent intervals.

To configure **telnetd** to have a **BUFFERSIZE** of 100 bytes and a **BUFFERTIMEOUT** of 100 ticks and the **TCP_DELAY** ON, the entry in **/etc/inetd.conf** would be:

```
telnet  stream tcp nowait root /usr/lbin/telnetd  telnetd -s100 \
-z100 -TCP_DELAY
```

DIAGNOSTICS

If any error is encountered by **telnetd** in establishing the connection, an error message is returned through the connection, after which the connection is closed and the server exits. Any errors generated by the login process or its descendents are passed through as ordinary data.

t

The following diagnostic messages are displayed by **telnetd**:

unable to allocate Telnet device

The server was unable to obtain a Telnet pseudo-terminal for use with the login process. Either all Telnet pseudo-terminals were in use or the **telm** driver has not been properly set up (see *tels*(7)).

Next step: Check the Telnet pseudo driver configuration of the host where **telnetd** is executing.

fork: No more processes

telnetd was unable to fork a process to handle the incoming connection.

Next step: Wait a period of time and try again. If this message persists, the server's host may have runaway processes that are using all the entries in the process table.

/usr/bin/login: ...

The login program could not be started via **exec***() for the reason indicated (see *exec*(2)).

WARNINGS

The terminal type name received from the remote client is converted to lowercase.

`telnetd` never sends TELNET *go ahead* commands.

AUTHOR

`telnetd` was developed by the University of California, Berkeley.

SEE ALSO

login(1), rlogin(1), telnet(1), inetd(1M), inetsvcs_sec(1M), ioctl(2), hosts(4), inetd.conf(4), inetd.sec(4), services(4), tels(7), stty(1), tty(7).

DOD MIL_STD 1782.

RFC 854 for the TELNET protocol specification.

NAME

telnetd - TELNET protocol server

SYNOPSIS

/usr/lbin/telnetd [**-b** [*bannerfile*]] [**-A**] [**-a** *authmode*] [**-t**]

DESCRIPTION

The **telnetd** daemon executes a server that supports the DARPA standard TELNET virtual terminal protocol. The Internet daemon (**inetd**) executes **telnetd** when it receives a service request at the port listed in the services data base for **telnet** using the **tcp** protocol (see *inetd*(1M) and *services*(4)).

telnetd operates by allocating a Telnet pseudo-terminal device (see *tels*(7)) for a client, then creating a login process which has the slave side of the Telnet pseudo-terminal as **stdin**, **stdout**, and **stderr**. **telnetd** manipulates the master side of the Telnet pseudo-terminal, implementing the TELNET protocol, and passing characters between the client and login process.

NOTE: **telnetd** no longer uses *pty*(7) devices; instead it uses special devices created for TELNET sessions only. For a full description, see *tels*(7).

When a TELNET session is started up, **telnetd** sends TELNET options to the client side, indicating a willingness to do *remote echo* of characters, to *suppress go ahead*, and to receive *terminal speed, terminal type,* and *authentication* information from the remote client. If the remote client is willing, the remote terminal type is propagated in the environment of the created login process. The Telnet pseudo-terminal allocated to the client is configured as a normal terminal with the exception of echoing characters (see *tty*(7)).

telnetd is willing to *do*: *echo*, *binary*, *suppress go ahead*, and *timing mark*.

telnetd is willing to have the remote client *do*: *binary*, *flow control*, *terminal speed*, *terminal type*, *suppress go ahead*, and *authentication*.

The flow control option permits applications running on a remote host to toggle the flow control on the local host. To toggle flow control for a **telnet** session programmatically, the application program must first call the **tcgetattr** function to get the current **termios** settings. For example,

```
tcgetattr(filedes, &termios_p)
```

Then, the **c_iflag** of the **termios** structure must have **IXON** set(reset) to enable(disable) flow control.

Finally, the **tcsetattr** function call can implement the change. For example,

```
tcsetattr(filedes, TCSANOW, &termios_p)
```

To toggle the flow control interactively, the user can issue a **stty** command using the input options **-ixon** to disable, or **ixon** to enable flow control. (see *stty(1)).*

The terminal speed option permits applications running on a remote host to obtain the terminal speed of the local host session using either *ioctl* or *stty*.

By default, the **telnet** server provides remote execution facilities with authentication based on Kerberos V5. (See *sis*(5).)

The **telnet** server also supports the TAC User ID (also known as the TAC Access Control System, or TACACS User ID) option, whereby users telneting to two or more consenting hosts may avoid going through a second login sequence. See the **-t** option below.

To start **telnetd** from the Internet daemon, the configuration file **/etc/inetd.conf** must contain an entry as follows:

```
telnet stream tcp nowait root /usr/lbin/telnetd telnetd
```

With this Kerberos version of **telnetd, inetd** can start **telnetd** with the following lines in **/etc/inetd.conf**. See the **-A** and **-a** options below.

```
telnet stream tcp nowait root /usr/lbin/telnetd telnetd -A
```

or

```
telnet stream tcp nowait root /usr/lbin/telnetd telnetd -a valid
```

telnet uses the same files as **rlogin** to verify participating systems and authorized users, **hosts.equiv** and **.rhosts**. (See *hosts.equiv*(4) and the *Managing Systems and Workgroups* manual for configuration details.)

Options

telnetd has the following options.

-b [*bannerfile*] Specify a file containing a custom banner. This option overrides the standard **telnetd** login banner. For example, to use **/etc/issue** as the login banner, have **inetd** start **telnetd** with the following lines in **/etc/inetd.conf** (\ provides line continuation):

```
telnet stream tcp nowait root /usr/lbin/telnetd \
    telnetd -b/etc/issue
```

If *bannerfile* is not specified, **telnetd** does not print a login banner.

-A Ensures that non-secure systems are denied access to the server. It overrides any value specified with the **-a** option except when *authmode* is **debug**. (See *sis*(5).)

-a *authmode* Specifies what mode is to be used for Kerberos authentication. (See *sis*(5).) Values for *authmode* are:

debug	Activates authentication debugging.
valid	Default value. Only allows connections when the remote user can provide valid Kerberos authentication information and is authorized to access the specified account.
none	Authentication information is not required. If no or insufficient Kerberos authentication information is provided, the *login*(1) program provides the necessary user verification.

-t Enable the TAC User ID option.

The system administrator can enable the TAC User ID option on servers designated as participating hosts by having **inetd** start **telnetd** with the **-t** option in **/etc/inetd.conf**:

```
telnet stream tcp nowait root /usr/lbin/telnetd telnetd -t
```

In order for the TAC User ID option to work as specified, the system administrator must assign to all authorized users of the option the same login name and unique user ID (UUID) on every participating system to which they are allowed TAC User ID access. These same UUIDs should not be assigned to non-authorized users.

Users cannot use the feature on systems where their local and remote UUIDs differ, but they can always use the normal **telnet** login sequence. Also, there may be a potential security breach where a user with one UUID may be able to gain entry to participating systems and accounts where that UUID is assigned to someone else, unless the above restrictions are followed.

A typical configuration may consist of one or more secure front-end systems and a network of participating hosts. Users who have successfully logged onto the front-end system may **telnet** directly to any participating system without being prompted for another login.

DIAGNOSTICS

If any error is encountered by **telnetd** in establishing the connection, an error message is returned through the connection, after which the connection is closed and the server exits. Any errors generated by the login process or its descendents are passed through as ordinary data.

Diagnostic messages displayed by **telnetd** are displayed below. Kerberos specific errors are listed in *sis*(5).

unable to allocate Telnet device

The server was unable to obtain a Telnet pseudo-terminal for use with the login process. Either all Telnet pseudo-terminals were in use or the **telm** driver has not been properly set up (see *tels*(7)).

Next step: Check the **tels** and **telm** configuration of the host where **telnetd** is executing.

fork: No more processes

telnetd was unable to fork a process to handle the incoming connection.

Next step: Wait a period of time and try again. If this message persists, the server's host may have runaway processes that are using all the entries in the process table.

/usr/bin/login: ...

The login program could not be started via **exec*()** for the reason indicated (see *exec*(2)).

WARNINGS

The terminal type name received from the remote client is converted to lowercase.

telnetd never sends TELNET *go ahead* commands.

AUTHOR

telnetd was developed by the University of California, Berkeley.

SEE ALSO

login(1), rlogin(1), stty(1), telnet(1), inetd(1M), inetsvcs_sec(1M), ioctl(2), hosts(4), inetd.conf(4), inetd.sec(4), services(4), sis(5), tels(7), tty(7).

Managing Systems and Workgroups.

DOD MIL_STD 1782.

RFC 854 for the TELNET protocol specification.

NAME

tftpd - trivial file transfer protocol server

SYNOPSIS

/usr/lbin/tftpd [**-R** *retran-seconds*] [**-T** *total-seconds*] [*path* ...]

DESCRIPTION

tftpd is a server that supports the Internet Trivial File Transfer Protocol (RFC783). The TFTP server operates at the port indicated in the **tftp** service description (see *services*(4)). The server is normally started by **inetd** using the **/etc/inetd.conf** file (see *inetd*(1M) and *inetd.conf*(4)).

The **-R** option specifies the per-packet retransmission timeout, in seconds. The default value is 5 seconds.

The **-T** option specifies the total retransmission timeout, in seconds. The default value is 25 seconds.

The *path* parameter has the following effects:

- **tftpd** operates in either of two modes or their combination. The first mode requires a defined home directory for the pseudo-user **tftp**, and looks for files relative to that path. The second mode requires one or more *paths* be specified on the command line, and allows access only to files whose paths match or begin with one of the command line specifications. The first mode is backward-compatible with previous releases of HP-UX and supports somewhat tighter security. The second mode is compatible with other vendors' implementations of **tftpd** and allows greater flexibility in accessing files.

- If no *path* is specified on the command line, **tftpd** requires an entry in the **/etc/passwd** database (see *passwd*(4)) for an account (pseudo-user) named **tftp**. The password field should be *****, the group membership should be **guest**, and the login shell should be **/usr/bin/false**. For example (assuming the guest group ID is 101):

  ```
  tftp:*:510:101:tftp server:/home/tftpdir:/usr/bin/false
  ```

  tftpd uses a call to **chroot()** to change its root directory to be the same as the home directory of the pseudo-user **tftp**. This restricts access by **tftp** clients to only those files found below the **tftp** home directory (see *chroot*(2)). Furthermore, **tftp** clients can only read files in that directory if they are readable by the pseudo-user **tftp**, and **tftp** clients can only write files in that directory if they exist and are writable by the pseudo-user **tftp**.

- If any *path* is specified on the command line, **tftpd** does not require that a pseudo-user named **tftp** exist in **/etc/passwd**. The specified *path*s control access to files by **tftp** clients. Each *path* is treated as being relative to **/** (not the **tftp** home directory), and can be either a directory or a file. **tftpd** disallows a client access to any file that does not match entirely or in its initial components one of the restriction *path*s. It also disallows access to any file path containing "**..**". However, an accessed file can be a symbolic link that points outside the set of restricted paths.

- If any *path* is specified on the command line and the **tftp** home directory is defined and is not **/**, **tftpd** first looks for a file relative to (under) the home directory. If the file is not found there, then **tftpd** looks for the file relative to **/** with path restrictions applied. Thus if two files with the same name can be found in both locations, **tftpd** accesses the one under **tftp**'s home directory.

Note that **inetd** allows continuation of command lines in **inetd.conf** by ending continued lines with a backlash.

Defining the **tftp** pseudo-user is strongly recommended even when *path*s are specified, because client access is further restricted to files that can be read and/or written by this pseudo-user. It is safe to set the **tftp** pseudo-user's home directory to **/** in this case.

DIAGNOSTICS

The following diagnostics are logged to the **syslogd** facility at the **err** log level (see *syslogd*(1M)).

No security mechanism exists

The pseudo-user **tftp** was not found in the password database (**/etc/passwd**), and **tftpd** was invoked without any *path* arguments.

Add or correct the entry for the pseudo-user **tftp** in the password database **/etc/passwd**. Or, add an access list (*path* arguments) to the **tftpd** arguments in the **inetd** configuration file **/etc/inetd.conf**. Reconfigure **inetd** with the command **inetd -c**.

`Unknown option` *option* **`ignored`**

An invalid option was specified in the **`tftpd`** arguments in the **`inetd`** configuration file **`/etc/inetd.conf`**.

Remove or correct the option. Restart **`inetd`** with the command **`inetd -c`**.

`Invalid total timeout` *value*

The value given for the **`-T`** option was not a number or was a negative number.

Correct the value given for the **`-T`** option. Reconfigure **`inetd`** with the command **`inetd -c`**.

`Invalid retransmission timeout` *value*

The value given for the **`-R`** option was not a number or was a negative number.

Correct the value given for the **`-R`** option. Reconfigure **`inetd`** with the command **`inetd -c`**.

*system call***`:`**

The named *system call* failed. See the corresponding manual entry for a description of the system call. The reason for the failure is explained in the error message appended to the system call.

WARNINGS

When invoked with no *path* arguments, **`tftpd`** cannot follow symbolic links that refer to paths outside of the home directory of the pseudo-user **`tftp`**, because it performs a **`chroot()`**.

AUTHOR

`tftpd` was developed by the University of California, Berkeley, and Hewlett-Packard.

SEE ALSO

tftp(1), inetd(1M), syslogd(1M), chroot(2), inetd.conf(4), passwd(4).

NAME

tic - terminfo compiler

SYNOPSIS

tic [**-v** [*n*]] [**-c**] *file* ...

DESCRIPTION

tic translates terminfo files from source format into the compiled format. Results are placed in the directory **/usr/share/lib/terminfo**.

-vn Specifies that (verbose) output be written to standard error trace information showing tic's progress. The optional integer **n** is a number from 1 to 10, inclusive, indicating the desired level of detail of information. If **n** is omitted, the default level is 1. If **n** is specified and greater than 1, the level of detail is increased.

-c Specifies to check only file for errors. Errors in **use=links** are not detected.

tic compiles all terminfo descriptions in the given files. When a **use=** field is discovered, **tic** searches first the current file, then the master file which is **./terminfo.src**.

If the environment variable **TERMINFO** is set, the results are placed in the location specified by **TERMINFO** instead of in **/usr/share/lib/terminfo**.

Limitations: total compiled entries cannot exceed 4096 bytes. The name field cannot exceed 128 bytes.

FILES

/usr/share/lib/terminfo/?/* compiled terminal capability data base

SEE ALSO

untic(1M), curses(3X), terminfo(4), captoinfo(1M).

BUGS

Instead of searching **./terminfo.src**, **tic** should check for an existing compiled entry.

STANDARDS CONFORMANCE

tic: SVID2, SVID3

NAME

tsm.lpadmin - add or remove a printer for use with *tsm*(1)

SYNOPSIS

/usr/tsm/bin/tsm.lpadmin -p *printer* **-m** *model*

/usr/tsm/bin/tsm.lpadmin -x *printer*

DESCRIPTION

tsm.lpadmin is used to add (or remove) a printer to the LP spooling system when the printer is connected to the system through a terminal running the Terminal Session Manager (see *tsm*(1)). **tsm.lpadmin** is a shell script that uses **lpadmin** in the normal way but also creates a named pipe to which LP output is directed (see *lpadmin*(1)). This named pipe is opened by TSM and data flowing from it is sent to the printer through the terminal.

Options

tsm.lpadmin recognizes the following options:

-p *printer*	Names a printer to be created with an associated pipe. If **-p** is used, **-m** must also be specified.
-m *model*	Selects a model interface program for *printer*. *model* is one of the model interface names supplied with the LP software (see the Models topic in the *lpadmin*(1)) manual entry. If **-m** is used, **-p** must also be specified.
-x *printer*	Removes *printer* from the LP system. No other options are allowed with **-x**.

Restrictions

To use **tsm.lpadmin** you must be user **lp** or **root**.

AUTHOR

tsm.lpadmin was developed by HP.

FILES

/var/spool/lp/tsm.pipes/*

SEE ALSO

lpadmin(1M), tsm(1).

NAME

ttsyncd - daemon to maintain the **nis+** password table in sync with the **nis+** trusted table

SYNOPSIS

/usr/lbin/ttsyncd [**-D**] [**-v**] [**-t** *synchour*] [**-i** *interval*]

DESCRIPTION

ttsyncd checks that each login name in the **nis+** password (**passwd**) table appears in the **nis+** trusted table. It will create a user entry in the trusted table for every user that exists in the password table and NOT in the trusted table. Each **nis+** user can potentially log in to an HP trusted system; thus, **ttsyncd** aids trusted systems by creating an entry in the trusted table before the **nis+** user logs in. In turn, the system administrator can modify global security attributes for each **nis+** user before they log in to a trusted system that is part of the **nis+** namespace.

Options

ttsyncd recognizes the following options:

-D Used for debugging **ttsyncd**. All output will go to the screen as well as the syslog.

-v Verbose mode used for debugging **ttsyncd**.

-t Initial daily sync time. The time is in military hour (0 through 23). For example,

-t10	sync time is 10 a.m
-t23	sync time is 11 p.m
-t0	midnight
default	sync once next sync at midnight

-i Sync interval. This value is allowed from 0 through 23. This is only in hours. For example,

-i0	sync continuously (see note below)
-i4	sync every 4 hours
-i20	sync every 20 hours
default	sync 24 hours

DIAGNOSTICS

If it is desired to administer Trusted mode centrally within a **nis+** namespace, **ttsyncd** should be running on every HP **nis+** server. Furthermore, **ttsyncd** can be started automatically at boot time by setting TTSYNCD=1 in **/etc/rc.config.d/comsec**. Additionally, **ttsyncd** will run only on a **nis+** server (root or master) and only if **nis+** is up and running.

EXAMPLES

Command format

Initial sync time at midnight and sync every 2 hours.

```
/usr/lbin/ttsyncd -t0 -i2
```

Initial sync time at 10 a.m and sync every 2 hours.

```
/usr/lbin/ttsyncd -t10 -i2
```

Initial sync time at midnight and sync every 4 hours.

```
/usr/lbin/ttsyncd -i4
```

Sync only at midnight.

```
/usr/lbin/ttsyncd
```

Continuously sync starting at midnight.

```
/usr/lbin/ttsyncd -i0
```

NOTE

If interval **-i0** is given, **ttsyncd** will keep checking the passwd table for change. If there is any change in the passwd table then it will do a sync. This is useful when trusted table needs to be sync'ed as soon passwd table is modified. The drawback is it will consume CPU time for checking the passwd table.

DEPENDENCIES

NIS+ (Network Information Name Service)

`ttsyncd` requires NIS+ to be configured and running. It should be run only on an **`nis+`** server. Moreover, **`ttsyncd`** will self-terminate if the password table does not exist.

AUTHOR

`ttsyncd` was developed by Hewlett Packard.

FILES

`/etc/rc.config.d/comsec`
`/etc/sbin/init.d/comsec`

SEE ALSO

getprpwent(3).

NAME

tunefs - tune up an existing HFS file system

SYNOPSIS

/usr/sbin/tunefs [**-A**] [**-v**] [**-a** *maxcontig*] [**-d** *rotdelay*] [**-e** *maxbpg*] [**-m** *minfree*] [**-r** *advanced read-ahead*] *special-device*

DESCRIPTION

The **tunefs** command is used to alter dynamic parameters that affect HFS file system layout policies. Parameters to be altered are specified by the options and arguments provided on the command line as described below.

tunefs affects how the file system blocks are laid out on the disk. The default *rotdelay* value set by the **newfs** and **mkfs** commands (see *newfs*(1M) and *mkfs*(1M)) is 0 milliseconds, causing file system blocks to be written and read consecutively. In general, this should be the optimal tuning, making the use of **tunefs -d** unnecessary.

Options

tunefs recognizes the following options and command-line arguments:

-a *maxcontig* Set the maximum number of contiguous blocks that will be laid out before forcing a rotational delay to *maxcontig* (see **-d** below). The default value is **1**, because most device drivers require one interrupt per disk transfer. For device drivers that can chain several buffers together in a single transfer, set *maxcontig* to the maximum chain length.

-d *rotdelay* *rotdelay* is the expected time (in milliseconds) to service a transfer completion interrupt and initiate a new transfer on the same disk. It is used to determine how much rotational spacing to place between successive blocks in a file.

-e *maxbpg* *maxbpg* specifies the maximum number of blocks any single file can allocate out of a cylinder group before it is forced to begin allocating blocks from another cylinder group. Typically this value is set to about one fourth of the total blocks in a cylinder group. The intent is to prevent any single file from using up all the blocks in a single cylinder group, thus degrading access times for all files subsequently allocated in that cylinder group. The effect of this limit is to cause large files to do long seeks more frequently than if they were allowed to allocate all the blocks in a cylinder group before seeking elsewhere. For file systems with exclusively large files, this parameter should be set higher.

-m *minfree* *minfree* specifies the percentage of space that is not available to normal users; i.e., the minimum free space threshold. The default value used is 10%. This value can be set to zero. If set to zero, throughput performance drops to as little as one-third of the efficiency expected when the threshold is set at 10%. Note that if *minfree* is raised above the current usage level, users cannot allocate files until enough files have been deleted to meet the new threshold requirement.

-r *advanced read-ahead*
Advanced read-ahead specifies whether the file system should use an advanced predictive read-ahead algorithm. The implementation requires more system resources in exchange for an advanced access pattern recognition. Patterns include forward sequential, backward sequential, forward strided, and backward strided. This value can be set to zero (disable) or one (enable). By default, a file system will have *advanced read-ahead* enabled when created.

-v (visual) Display current values contained in the primary super-block to standard output.

-A (all) Modify redundant super-blocks as well as the primary super-block as stipulated by the configuration options and arguments.

special-device is the name of the file system to be tuned. It is either a block or character special file if the file system is not mounted, or a block special file if the file system is mounted.

WARNINGS

Root file system tuning is normally done during initial system software installation. Tuning the root file system after installation has little useful effect because so many files have already been written.

You can tune a file system, but you can't tune a fish.

AUTHOR

`tunefs` was developed by the University of California, Berkeley.

SEE ALSO

dumpfs(1M), mkfs(1M), newfs(1M), fs(4).

NAME

udpublickey - update the publickey database file and the NIS map

SYNOPSIS

udpublickey

Remarks

The Network Information Service (NIS) was formerly known as Yellow Pages (yp). Although the name has changed, the functionality of the service remains the same.

DESCRIPTION

udpublickey is executed from the *updaters*(1M) makefile when either **newkey** or **rpc.ypupdated** updates the **/etc/publickey** database file.

udpublickey receives the following information from **newkey** or **rpc.ypupdated:**

Requestor's name (a string)

Type of update

Number of bytes in key

Key

Number of bytes in data

Data

After receiving this information, **udpublickey** attempts to update the publickey database file, **/etc/publickey**.

If the update is successful, **udpublickey** makes the NIS map, **publickey.byname**.

If the update is completely successful, **udpublickey** exits with a zero (0) status; otherwise **udpublickey** exits with a valid NIS error.

This command should not be run interactively.

AUTHOR

udpublickey was developed by Sun Microsystems, Inc.

FILES

/etc/publickey

SEE ALSO

newkey(1M), rpc.ypupdated(1M), updaters(1M), publickey(4).

NAME

untic - terminfo de-compiler

SYNOPSIS

untic [*term*] [**-f** *file*]

DESCRIPTION

untic translates a terminfo file from the compiled format into the source format. If the environment variable **TERMINFO** is set to a path name, **untic** checks for a compiled terminfo description of the terminal under the path specified by **TERMINFO** before checking **/usr/share/lib/terminfo**. Otherwise, only **/usr/share/lib/terminfo** is checked.

Normally **untic** uses the terminal type obtained from the **TERM** environment variable. With the *term* (terminal type) argument, however, the user can specify the terminal type used.

With the *file* argument the user can specify the file used for translation. This option bypasses the use of the **TERM** and **TERMINFO** environment variables.

untic sends the de-compiled terminfo description result to standard output.

AUTHOR

untic was developed by HP.

FILES

/usr/share/lib/terminfo/?/*	compiled terminal capability data base

SEE ALSO

tic(1M), curses(3X), terminfo(4).

NAME

update-ux - updates the HP-UX operating system from new HP-UX media

SYNOPSIS

update-ux -s *source_location* [**-?**] [**-a** *bits*] [**-n**|**-y**] [**-i**] [**-x** *option=value*] [*sw_spec*]

DESCRIPTION

The **update-ux** command updates the HP-UX operating system to a newer version.

Use **update-ux** when updating the operating system (OS), changing the OS word width, and installing or changing operating environments (OEs). **update-ux** works only with source depots containing the OS and OEs such as HP-UX 11i OE CDs. When using other media as source, use *swinstall*(1M) instead.

update-ux will always attempt to update the OS. These bundles are installed by default from HP-UX 11i OE CDs:

HPUXBase32 or HPUXBase64
HPUXBaseAux
FibrChanl-00 (64-bit OS only)
GigEther-00
RAID-00 (64-bit OS only)
FDDI-00 (32-bit OS only)
OnlineDiag
CDE language bundle based on installed CDE.
Other network drivers based on installed software.

To list all product bundles available on the CD mounted at **/cdrom**, enter:

```
/usr/sbin/swlist -s /cdrom
```

If you are changing OEs within the same OS version, **update-ux** will attempt to update the OS. However, only software that has not yet been installed will be installed. Software already on the target system will be deselected during the analysis phase.

If the current OS is 10.20 or 11.00, first install **update-ux** onto the existing system. For example:

```
swinstall -r -s source_location SW-DIST.SD-UPDATE \@ \
/var/adm/sw/update-ux.root 2>/dev/null
```

where *source_location* is the full path to a depot containing the 11i SW-DIST product.

The redirection in this command is recommended when updating from 10.20 to avoid inconsequential warning and error messages. This is because the 10.20 version of swinstall does not recognize many of the new keywords added to the newer products in this depot. Since these errors and warnings can be ignored, be sure to add the redirection of standard error.

Options

update-ux supports these options:

-s *source_location* Specify the source location. Possible locations are a local directory, a mounted CD-ROM containing an SD depot, or a remote machine and depot combination. All paths used in the *source_location* must be absolute paths. If *source_location* is a CD-ROM, **update-ux** expects to install from two CDs and will prompt for the second CD. If *source_location* is a remote machine and depot combination, the remote machine should be specified first, followed by the absolute path to the remote depot, separated by a colon with no spaces; for example: **swperf:/var/spool/sw**

-? Print the usage statement.

-a *bits* Specify 32- or 64-bit OS architecture. Omitting **-a 32** or **-a 64** defaults to the currently-set architecture (only 64-bit OS is supported on B-, C- and J-class systems).

-n|**-y** By default, **update-ux** pauses when an error is detected to ask if it should continue. Specify "no" (**-n**) or "yes" (**-y**) to tell **update-ux** whether to continue when a error message is issued. Use **-n** (or omit this option) at first to have **update-ux** abort when an error is encountered. Then review the error logged in **/var/adm/sw/swagent.log**. When you have reviewed/resolved

u

	the errors, use **`-y`** to have **`update-ux`** ignore any remaining errors.
`-i`	Start the **`swinstall`** interactive interface. For more information, see the *swinstall*(1M) manpage.
`-x` *option*=*value*	Set *swinstall*(1M) options. **`swinstall -p`** (preview) is not supported. For more information, see the *swinstall*(1M) manpage.
sw_spec	Specify optional software selection such as an OE or network-driver bundle name.

RETURN VALUES

The **`update-ux`** command returns a value when it is not successful:

1	Error during execution; update aborted.
2	Update aborted per user request (**`-n`** option) or from keyboard.

DIAGNOSTICS

Standard Output:

An **`update-ux`** session writes messages for significant events, including:

- Begin- and end-session messages.
- Major task messages.

Logging:

Errors are recorded in: **`/var/adm/sw/swagent.log`**.

EXAMPLES

To update from the OS depot on an HP-UX 11i OE CD-ROM mounted at **`/cdrom`**, enter:

```
/usr/sbin/update-ux -s /cdrom HPUX11i-OE
```

That example updates the OS and installs the HP-UX 11i Operating Environment (OE). You will be prompted to replace OE CD1 with CD2.

To update the OS from **`/cdrom`**, but not install an OE and ignore any error messages, enter:

```
/usr/sbin/update-ux -s /cdrom -y
```

To update the OS and also update any software on the target system with new versions found in the source depot at **`/cdrom`**, enter:

```
/usr/sbin/update-ux -s /cdrom -x match_target=true -a 64
```

That example also ensures that the 64-bit OS is installed on a target system that can run either 32- or 64-bit OS.

AUTHOR

`update-ux` was developed by HP.

FILES

`/usr/sbin/update-ux`	The **`update-ux`** command.
`/var/adm/sw/swagent.log`	The log file.

SEE ALSO

sd(4), sd(5), swinstall(1M), install-sd(1M).

These manuals are available on the HP-UX 11i Instant Information CD and at **`http://docs.hp.com/`**:

- *HP-UX 11i Installation and Update Guide*
- *Software Distributor Administration Guide*
- *Managing Systems and Workgroups*
- *Managing Superdome Complexes*

NAME

updaters - configuration file for NIS updating

SYNOPSIS

updaters

Remarks

The Network Information Service (NIS) was formerly known as Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

DESCRIPTION

updaters is a makefile used for updating the Network Information Service (NIS) databases. Databases can be updated only if the network is secure, that is, only if there is a NIS **publickey** database (**publickey.byname**). The default **updaters** script will update only the **publickey.byname** map.

An entry in the file is a make target for a particular NIS database. For example, if you wanted to add **passwd.byname** to this script, you would create a make target named passwd.byname with the command to update that database. See *udpublickey*(1M).

The information necessary to make the update is passed to the update command through standard input. The information passed is described below. All items are followed by a NEW LINE except for Actual bytes of key and Actual bytes of data.

Network name of client wishing to make the update (a string)

Kind of update (an integer)

Number of bytes in key (an integer)

Actual bytes of key

Number of bytes in data (an integer)

Actual bytes of data

After receiving this information through standard input, the command to update the particular database should decide whether the user is allowed to make the requested change.

If not, the command should exit with the status **YPERR_ACCESS**.

If the user is allowed to make the change, the command should make the change and exit with a status of zero.

If there are any errors that may prevent the updater from making the change, it should exit with the status that matches a valid NIS error code described in **<rpcsvc/ypclnt.h>**.

AUTHOR

updaters was developed by Sun Microsystems, Inc.

SEE ALSO

make(1), newkey(1M), rpc.ypupdated(1M), udpublickey(1M), publickey(4).

NAME

ups_mond - HP PowerTrust Uninterruptible Power System monitor daemon

SYNOPSIS

/usr/lbin/ups_mond [**-f** *configfile*] [**-s**]

DESCRIPTION

When it detects a loss of AC power for a period of time exceeding a configured limit, **ups_mond** ensures file system integrity by shutting down HP-UX. To do this, **ups_mond** uses the device special files specified in its configuration file (**/etc/ups_conf** by default) to monitor the state of each HP PowerTrust Uninterruptible Power System (UPS) attached to the system.

Use the **-f** option to specify a configuration file other than **/etc/ups_conf**. See *ups_conf*(4) for a description of the configuration file format.

By default, **ups_mond** is locked into memory (see *plock*(2)). That is, **ups_mond** is not swappable. Although extreme caution is required, you can make **ups_mond** swappable if all swap disks are powered by an uninterruptible power system (assured to have power when the primary power source fails). To make **ups_mond** swappable, use the **-s** option.

ups_mond is started by **init** (see *init*(1M)) by means of an entry in the file **/etc/inittab** (see *inittab*(4)). The **inittab** entry uses the **respawn** option to automatically restart **ups_mond** if **ups_mond** is terminated by the **kill** command (see *kill*(1)). This entry should follow the entry:

```
sqnc::wait:/sbin/rc </dev/console >/dev/console 2>&1 # system initialization
```

so that **ups_mond** is started after the system logging daemon (**syslogd**). It should also be run with real-time priority to assure its execution (see *rtprio*(1)).

ups_mond logs messages, and when appropriate invokes **/usr/sbin/shutdown** using the **-h** option, or **/usr/sbin/reboot**. For each configured UPS, **ups_mond** can be instructed (in **/etc/ups_conf**) to log messages only, without taking **shutdown** or **reboot** action. See MSG_ONLY in *ups_conf*(4). By default **ups_mond** performs the **shutdown** and **reboot** actions.

Note that when the shutdown is performed, UPSs that have lost AC line voltage will be turned off once the *shutdown_timeout_mins* time has expired (see *ups_conf*(4)). By default the system will power on when the AC line voltage is restored. The *kill_after_shutdown* line can be added to **ups_conf** to tell the UPS not to come back up when AC line voltage is restored (see *ups_conf*(4)).

ups_mond uses the **syslog** message logging facility to log these occurrences (see *syslog*(3C)). Messages are written to the console if **ups_mond** is unable to send them to **syslogd**. Critical messages (see DIAGNOSTICS section) are also sent to the console.

RETURN VALUE

ups_mond returns the following values:

zero (0)	Successful Completion
non-zero	Error encountered. See ERRORS below.

EXAMPLES

The entry in **/etc/inittab** should be similar to this:

```
ups::respawn:rtprio 0 /usr/lbin/ups_mond -f /etc/ups_conf
```

DIAGNOSTICS

Messages resulting from normal operation:

UPS Monitor daemon starting; using configuration file <*configfilename***>.**

UPS <*tty special file name***> OK: AC Power back on.**

AC Power to all recognized, system critical UPS's OK! System will not shutdown.

Messages resulting in exit of daemon:

usage: ups_mond [-f configfile].

cannot exec /usr/lbin/ups_mond -f <*configfilename***> -e ups_monchild due to <***error***>.**

permission denied; must be superuser.

exiting; unable to lock process in memory: <*errno***>.**

aborted, configfile <*configfilename***> open received error: <***errno***>.**

aborted, configfile <*configfilename***> fseek error: <***errno***>.**

aborted, malloc error: <*errno***>.**

terminated by signal <*decimal value of signal***>.**

Messages for which **shutdown** might be run (depends on UPS configuration):

UPS <*tty special file name***> AC POWER FAILURE - running on UPS battery.**

If power is not returned within previously configured time period, your system will automatically go to graceful shutdown.

If power is not returned within previously configured time period, your system will automatically go to graceful shutdown. System will not come back up after shutdown.

Messages for which **reboot** might be run (depends on UPS configuration):

UPS <*tty special file name***> battery low.**

UPS <*tty special file name***> no output - either switch setting wrong on UPS or bad UPS.**

UPS <*tty special file name***> failed - requires repair.**

UPS <*tty special file name***> current overload; UPS turned itself off - either UPS bad or too many devices connected.**

UPS <*tty special file name***> ambient temperature too high; UPS turned itself off - reduce heat in area.**

UPS <*tty special file name***> output voltage too high; UPS turned itself off - requires repair.**

UPS <*tty special file name***> output voltage too low; UPS turned itself off - requires repair.**

cannot exec shutdown due to <*errno***>.**

The above messages are followed by the following message:

reboot -halt invoked due to UPS error cited in previous syslog message.

Messages that are only logged (no **shutdown**/**reboot** action is taken):

warning - no upstty: UPS's found in configfile <*configfilename***>; daemon running for no purpose.**

warning - shutdown delay or shutdown timeout parameter in configfile <*configfilename***> missing or not greater than zero; using default.**

UPS <*tty special file name***> in bypass-mode; no AC Power-loss protection.**

UPS <*tty special file name***> interrupted, but read of ups status failed - possible UPS hardware problem.**

upstty <*tty special file name***> failed open: <***errno***>; ignoring that tty and continuing.**

UPS <*tty special file name***> ioctl(TCGETA) failed: <***errno***>; ignoring that UPS.**

UPS <*tty special file name***> ioctl(TCSETAF) failed <***errno***>; ignoring that UPS.**

UPS <*tty special file name***> line too noisy; ignoring that UPS.**

UPS <*tty special file name***> could not enable; loss of power would not be detectable.**

UPS <*tty special file name***> read failed: <***errno***>; Uninterruptible Power Supply has not been connected correctly; loss of power would not be**

detectable.

UPS <*tty special file name***> write failed: <***errno***>; ignoring that UPS.**

UPS <*tty special file name***> read of status received ILLEGAL CMD or NOISY LINE.**

UPS <*tty special file name***> read of status received <***number***> bytes of unexpected data (octal: <***octal returned***>): <***string returned***>.**

UPS <*tty special file name***> read of status failed: <***errno***>.**

UPS <*tty special file name***> write failed: <***errno***>.**

UPS <*tty special file name***> turned-off Failure Alarm.**

UPS <*tty special file name***> turned-off Inverter Failure Alarm.**

UPS <*tty special file name***> turned-off No Battery Alarm.**

UPS <*tty special file name***> turned-off Battery Charger Fault Alarm.**

UPS <*tty special file name***> turned-off Current Overload Alarm.**

UPS <*tty special file name***> turned-off High Ambient Temperature Alarm.**

UPS <*tty special file name***> turned-off Battery Failure Alarm.**

UPS <*tty special file name***> turned-off High Battery Voltage Alarm.**

UPS <*tty special file name***> turned-off Low Battery Voltage Alarm.**

UPS <*tty special file name***> turned-off High Output Voltage Alarm.**

UPS <*tty special file name***> turned-off Low Output Voltage Alarm.**

UPS <*tty special file name***> Inverter Failure requires repair.**

UPS <*tty special file name***> No Battery - ensure UPS battery installed.**

UPS <*tty special file name***> Battery Charger Fault- requires repair.**

UPS <*tty special file name***> Current Overload - either UPS bad or too many devices connected.**

UPS <*tty special file name***> High Ambient Temperature- reduce area temperature.**

UPS <*tty special file name***> Battery Failure- requires repair.**

UPS <*tty special file name***> High Battery Voltage - requires repair.**

UPS <*tty special file name***> Low Battery Voltage - requires repair.**

UPS <*tty special file name***> UNKNOWN status/alarm <***hex number***> - may require repair.**

write to UPS <*tty special file name***> of command <***cmd string***> Failed: <***errno***>.**

read from UPS <*tty special file name***> after sending command <cmd string> to it failed; <***errno***>.**

UPS <*tty special file name***> could not execute command <***cmd string***>; returned (octal: <***octal returned***>): <***string returned***> - possible bad signal cable.**

Messages relating to Timer Controlled Power On and Off:

Timer Controlled On/Off information invalid; ignored.

mknod error: <*errno***> for Timed On/Off fifo file /var/tmp/timed_off; continuing without.**

open error: <*errno***> for Timed On/Off fifo file /var/tmp/timed_off; continuing without.**

Timer Controlled On value exceeds UPS <*tty special file name***> maximum. The maximum value of <***maximum supported decimal value***> will be used for this UPS.**

u

ERRORS

ups_mond returns the following error values:

EINVAL	**ups_mond** encountered an incorrect parameter.
EPERM	Insufficient privileges. **ups_mond** must be started by a superuser.
EINTR	**ups_mond** was interrupted (terminated) by **signal**() or **kill**(). See *signal*(2) and *kill*(1).
one (1)	For all other error conditions.

FILES

/dev/tty*
/etc/ups_conf
/var/tmp/timed_off
/var/adm/syslog/syslog.log

SEE ALSO

kill(1), init(1M), plock(2), signal(2), syslog(3C), inittab(4), ups_conf(4).

NAME

useradd - add a new user login to the system

SYNOPSIS

useradd [**-u** *uid* [**-o**]] [**-g** *group*] [**-G** *group* [**,** *group* ...]] [**-d** *dir*] [**-s** *shell*] [**-c** *comment*] [**-m** [**-k** *skel_dir*]] [**-f** *inactive*] [**-e** *expire*] *login*

useradd -D [**-g** *group*] [**-b** *base_dir*] [**-f** *inactive*] [**-e** *expire*]

DESCRIPTION

The **useradd** command creates a user login on the system by adding the appropriate entry to the **/etc/passwd** file and any security files, modifying the **/etc/group** file as necessary, creating a home directory, and copying the appropriate default files into the home directory. The new login remains locked until the **passwd** (see *passwd*(1)) command is invoked.

New Behavior

login will not be added to the primary group entry in the **/etc/group** file, even if the primary group is specified in the command line. However, the *login* is added to the corresponding supplemental group in **/etc/group** file.

Options

The **useradd** command supports the following options:

-u *uid*	Specifies the UID for the new user. *uid* must be a non-negative decimal integer less than **MAXUID** as it is defined in the <**param.h**> header file. *uid* defaults to the next available unique number above the maximum currently assigned number. UIDs from 0-99 are reserved.
-o	Allows the UID to be non-unique (i.e., a duplicate).
-g *group*	Specifies the integer group ID or character string name of an existing group. This defines the primary group membership of the new login. The default for this option can be reset by invoking **useradd -D -g** *group*.
-G *group*	Specifies the integer group ID or character string name of an existing group. This defines the supplemental group memberships of the new login. Multiple groups may be specified as a comma separated list. Duplicates within *group* with the **-g** and **-G** options are ignored.
-d *dir*	Specifies the home directory of the new login. It defaults to *base_dir/login*, where *login* is the new login and *base_dir* is the base directory for new login home directories.
-s *shell*	Specifies the full pathname of the new login shell. The default is an empty field, which causes the system to use **/sbin/sh** as the login shell. The value of *shell* must be a valid executable file.
-c *comment*	Specifies the comment field present in the **/etc/passwd** entry for this login. This can be any text string. A short description of the new login is suggested for this field.
-m	Creates the home directory for the new login if it does not exist. If the home directory exists, the directory must have read and execute permission by *group*, where *group* is the primary group of the new login.
-k *skel_dir*	Specifies the skeleton directory that contains information that can be copied to the new login's home directory. This directory must exist. The system provides a skeleton directory, **/etc/skel**, that can be used for this purpose.
-f *inactive*	Specifies the maximum number of days of continuous inactivity of the login before the login is declared invalid. Normal values are positive integers, while a value of −1 defeats this status.
-e *expire*	Specifies the date on which this login can no longer be used. After *expire*, no user will be able to access this login. This option is used to create temporary logins. *expire*, which is a date, may be typed in any format, except a Julian date. For example, a date may be entered in either of the following formats:

```
July 13, 1993
7/13/93
```

u

A value of `''''` defeats the expired date status.

-D Manages the defaults for various options. When **useradd** is invoked with this option only, the default values for *group*, *base_dir*, *skel_dir*, *shell*, *inactive*, and *expire* are displayed. Invoking **useradd** with this option and other allowed options sets the default values for those options.

-b *base_dir* Specifies the default base directory for the system. If **-d** *dir* is not specified, *base_dir* is concatenated with the new login name to define the path of the new home directory. *base_dir* must exist.

The **useradd** command may be used with the *login* argument, where *login* is the new login name, specified as a string of printable characters. It may not contain a colon (**:**) or a newline (**\n**).

Unless enhanced security is installed (see *pwconv*(1M)), the **-e** and **-f** options are not supported and will return an error.

Networking Features

NIS

This command is aware of NIS user and group entries. Only local users and groups may be modified with this command. Attempts to modify an NIS user or group will result in an error. NIS users and groups must be administered from the NIS server. NIS users are checked when verifying uniqueness of the new UID or new user name, which may result in the error

login *x* **not unique**

(return value 9), or the error

UID # is not unique (when -o is not used)

(return value 4) even though the user or UID is not present in the local **/etc/passwd** file. The error

Cannot modify /etc/group file, /etc/passwd was modified

(return value 10) is returned if a group specified with either the **-g** option or the **-G** option is an NIS group (see *group*(4)).

NFS

Errors may occur with the **-m** or **-k** options if the indicated directory is within an NFS mounted file system that does not allow root privileges across the NFS mount, and the directory or files within the directory do not have sufficient permissions.

RETURN VALUE

useradd exits with one of the following values:

0 Successful completion.

2 Invalid command syntax.

3 Invalid argument supplied to an option.

4 *uid* is not unique (when **-o** is not used).

6 The *group* specified with the **-g** option does not exist.

9 *login* is not unique.

10 Cannot modify the **/etc/group** file. The login was added to the **/etc/passwd** file, but not to the **/etc/group** file.

12 Unable to create the home directory (while using the **-m** option) or unable to complete the copy of *skel_dir* to the new home directory.

13 Unable to open **/etc/ptmp** file or **/etc/default** file, or **/etc/passwd** file is non-existent.

14 **/etc/passwd**, or **/etc/ptmp**, or **/etc/default** file busy. Another command may be modifying the **/etc/passwd** file.

16 Cannot add the entry into the **/etc/passwd** file.

EXAMPLES

Add the user **otto** to the system with all of the default attributes.

```
useradd otto
```

Add the user **otto** to the system with a UID of **222** and a primary group of **staff**.

```
useradd -u 222 -g staff otto
```

List the defaults for the primary group, base directory, inactivity timeout, and skeleton directory.

```
useradd -D
```

Change the default primary group to **staff**.

```
useradd -D -g staff
```

WARNINGS

A directory can be shared between the users belonging to the same group. If the home directory is in the unshared mode and a new user is allocated to that directory then it will be put into the shared mode by setting the permissions of that directory to **775** (i.e. includes the write permissions to the group as well). Also, the directory which will be shared should have **read** and **execute** permissions for the group. Otherwise, **useradd** will report an error.

As many users may try to write the **/etc/passwd** file simultaneously, a passwd locking mechanism was deviced. If this locking fails after subsequent retrying, **useradd** terminates.

A group entry in the **/etc/group** file can have maximum of **LINE_MAX** bytes. If a user is added to a group that has reached **LINE_MAX** limit, another entry of the same group is created to which the new user is added. A warning message is also issued.

FILES

/etc/passwd
/etc/skel
/etc/group
/etc/ptmp

SEE ALSO

passwd(1), users(1), groupadd(1M), groupdel(1M), groupmod(1M), logins(1M), userdel(1M), usermod(1M), group(4).

STANDARDS COMPLIANCE

useradd: SVID3

NAME

userdel - delete a user login from the system

SYNOPSIS

userdel [**-r**] *login*

DESCRIPTION

The **userdel** command deletes a user login from the system by modifying the appropriate login related files.

The **userdel** command requires the *login* argument. *login* is the name to be deleted, specified as a string of printable characters. It may not contain a colon (:) or a newline (\n).

Option

The following option is available to this command:

-r The home directory of *login* is removed from the system. This directory must exist. Following the successful execution of this command, none of the files and directories under the home directory will be available.

If a user is deleted and the home directory is shared by others, then this directory is not deleted even with the **-r** option.

In the event where a directory is shared by users of the same group and the owner of that directory is deleted, then the ownership of that directory is propagated to the next user who is sharing that directory. The new owner is determined by looking at the order in which the users sharing this directory are added to the **/etc/passwd** file. If there is only one user remaining then the directory is brought back to unshared mode by resetting the permissions to **755** from **775**.

NETWORKING FEATURES

NIS

This command is aware of NIS user and group entries. Only local users and groups may be deleted or modified with this command. Attempts to delete or modify NIS users or groups will result in an error. NIS users and groups must be administered from the NIS server. The **userdel** command may fail with the error

login *x* **does not exist**

(return value 6) if the user specified is an NIS user (see *passwd*(4)). The error

Cannot modify /etc/group file, /etc/passwd was modified

(return value 10) is returned if a local user belongs to an NIS group (see *group*(4)).

NFS

Errors may occur with the **-r** option if the affected directory is within an NFS mounted file system that does not allow root privileges across the NFS mount, and the directory or files within the directory do not have sufficient permissions.

RETURN VALUES

userdel exits with one of the following values:

0	Successful completion.
2	Invalid command syntax.
3	Invalid argument supplied to an option.
6	The *login* to be removed does not exist.
8	The *login* to be removed is in use.
10	Cannot modify the **/etc/group** file, but the login was removed from the **/etc/passwd** file.
12	Unable to remove or modify the home directory.
13	Unable to open **/etc/ptmp** file or **/etc/passwd** file is non-existent.
14	**/etc/passwd** file or **/etc/ptmp** file busy. Another command may be modifying the **/etc/passwd** file.

17 Cannot delete entry from **/etc/passwd** file.

EXAMPLES

Remove the user **otto** from the system:

```
userdel otto
```

Remove the user **bob** from the system and delete **bob**'s home directory from the system:

```
userdel -r bob
```

WARNINGS

As many users may try to write the **/etc/passwd** file simultaneously, a passwd locking mechanism was deviced. If this locking fails after subsequent retrying, **userdel** terminates.

FILES

/etc/passwd
/etc/group
/etc/ptmp

SEE ALSO

passwd(1), users(1), groupadd(1M), groupdel(1M), groupmod(1M), logins(1M), useradd(1M), usermod(1M), group(4), passwd(4),

STANDARDS COMPLIANCE

userdel: SVID3

NAME

usermod - modify a user login on the system

SYNOPSIS

usermod [**-u** *uid* [**-o**]] [**-g** *group*] [**-G** *group* [**,** *group*...]] [**-d** *dir* [**-m**]] [**-s** *shell*] [**-c** *comment*] [**-f** *inactive*] [**-l** *new_logname*] [**-e** *expire*] *login*

DESCRIPTION

The **usermod** command modifies a user login on the system by changing the appropriate login related files.

The **usermod** command requires the *login* argument. *login* is a new login name, specified as a string of printable characters. It may not contain a colon (**:**) or a newline (**\n**).

New Behavior

If the primary group of a user is modified, then the user name is not added to the primary group entry in **/etc/group** file. However, if **-G** option is specified the user is added to the corresponding supplemental group.

Options

The **usermod** command supports the following options:

-u *uid*	Specifies the UID for the new user. *uid* must be a non-negative decimal integer less than **MAXUID** as it is defined in the <**param.h**> header file.
-o	Allows the UID to be non-unique (i.e., a duplicate).
-g *group*	Specifies the integer group ID or character string name of an existing group. This redefines the primary group membership of the new login.
-G *group*	Specifies the integer group ID or character string name of an existing group. This redefines the supplemental group memberships of the new login. Duplicates within *group* with the **-g** and **-G** options are ignored.
-d *dir*	Specifies the new home directory of the login. It defaults to *base_dir/login*, where *login* is the new login and *base_dir* is the base directory for new login home directories.
-m	Move the user's home directory to the directory specified with the **-d** option. If the home directory exists, the directory must have read and execute permission by *group*, where *group* is the primary group of the login.
-s *shell*	Specifies the full pathname of the login shell. The value of *shell* must be a valid executable file.
-c *comment*	Specifies the comment field present in the **/etc/passwd** entry of this login. This can be any text string. A short description of the new login is suggested for this field.
-f *inactive*	Specifies the maximum number of days of continuous inactivity of the login before the login is declared invalid. Normal values are positive integers, while a value of −1 defeats this status.
-l *new_logname*	Specifies the new login name for the user. It consists of a string of printable characters that does not contain a colon (**:**) or a newline (**\n**).
-e *expire*	Specifies the date on which this login can no longer be used. After *expire*, no user will be able to access this login. This option is used to create temporary logins. *expire*, which is a date, may be typed in any desired format, except a Julian date. For example, a date may be entered as either of the following:

```
July 13, 1993
7/13/93
```

A value of **''''** defeats the expired date status.

Unless enhanced security is installed (see *pwconv*(1M)), the **-e** and **-f** options are not supported and will return an error.

A directory can be shared between the users belonging to the same group. If the home directory is in unshared mode and a new user is allocated to that directory, then it will be put into shared mode by setting the permissions of that directory to **775** (i.e., includes the write permissions to the group as well). Also, the directory which will be shared should have read and execute permissions for the group.

In the event where a directory is shared by users of the same group and the owner of that directory is modified, then the ownership of that directory is propagated to the next user who is sharing that directory. The new owner is determined by looking at the order in which the users sharing this directory are added to the **/etc/passwd** file. If there is only one user remaining then the directory is brought back to unshared mode by resetting the permissions to **755** from **775**.

If a directory is shared by users, then one cannot change the primary group of any of these users unless the home directory of that user is also changed.

Networking Features

NIS

The **usermod** command is aware of NIS user and group entries. Only local users and groups may be modified with this command. Attempts to modify an NIS user or group will result in an error. NIS users and groups must be administered from the NIS server. This command may fail with the error

login *x* **does not exist**

(return value 6) if the user specified is an NIS user (see *passwd*(4)). However, NIS users are checked when verifying uniqueness of the new UID or the new user name. Also, the error

Cannot modify /etc/group file, /etc/passwd was modified

(return value 10) may be returned if a group specified with either the **-g** option or the **-G** option is an NIS group (see *group*(4)).

NFS

Errors may occur with the **-m** option if either the source or the target directory is within an NFS mounted file system that does not allow root privileges across the NFS mount and the directory or files within the directory do not have sufficient permissions.

RETURN VALUE

usermod exits with one of the following values:

0	Successful completion.
2	Invalid command syntax.
3	Invalid argument supplied to an option.
4	*uid* is not unique (when **-o** is not used).
6	The *login* to be modified or the *group* specified with the **-g** option does not exist.
8	The *login* to be modified is in use.
9	*new_logname* is not unique.
10	Cannot modify the **/etc/group** file. The other parts of the update request will be performed.
11	There is insufficient space to move the home directory (with the **-m** option). The other parts of the update request will be performed.
12	Unable to complete the move of the home directory to the new home directory.
13	Unable to open **/etc/ptmp** file, or **/etc/passwd** file is non-existent.
14	**/etc/passwd** file or **/etc/ptmp** file busy. Another command may be modifying the **/etc/passwd** file.
15	Cannot modify the entry in the **/etc/passwd** file.

EXAMPLES

Change **otto**'s primary group to **staff**.

```
usermod -g staff otto
```

Change **otto**'s user ID to **333** and change the login name to **bob**.

u

```
usermod -u 333 -l bob otto
```

WARNINGS

As many users may try to write the **/etc/passwd** file simultaneously, a passwd locking mechanism was devised. If this locking fails after subsequent retrying, **usermod** terminates.

While modifying the user login, the username is not added to the primary group entry in the **/etc/group** file. If a supplemental group is specified, the user is added to the supplemental group. If the size of a group entry in **/etc/group** file exceeds **LINE_MAX** limit, a new entry of the same group is created and a warning message is issued.

FILES

/etc/passwd
/etc/group
/etc/ptmp

SEE ALSO

passwd(1), users(1), groupadd(1M), groupdel(1M), groupmod(1M), logins(1M), useradd(1M), userdel(1M), group(4).

STANDARDS COMPLIANCE

usermod: SVID3

NAME

uucheck - check the uucp directories and permissions file

SYNOPSIS

/usr/lbin/uucp/uucheck [**-v**] [**-x** *debug_level*]

DESCRIPTION

uucheck checks for the presence of the files and directories required by **uucp** (see *uucp*(1)). **uucheck** is executed from the UUCP makefile before the installation occurs. **uucheck** also checks for various obvious errors in the **/etc/uucp/Permissions** file.

Options

uucheck recognizes the following options and command-line arguments:

-v (verbose) Print a detailed explanation of how **uucp** programs will interpret the **Permissions** file.

-x *debug_level*
Debug. *debug_level* is a single digit; the higher the number, the more detail returned.

Note that **uucheck** can only be used by the super-user or **uucp**.

FILES

/etc/uucp/Systems
/etc/uucp/Permissions
/etc/uucp/Devices
/etc/uucp/Maxuuxqts
/etc/uucp/Maxuuscheds
/var/spool/uucp/*
/var/spool/locks/LCK*
/var/spool/uucppublic/*

SEE ALSO

uucp(1), uustat(1), uux(1), uucico(1M), uusched(1M).

Tim O'Reilly and Grace Todino,
Managing UUCP and Usenet, O'Reilly & Associates, Inc. USA.

Grace Todino and Dale Dougherty,
Using UUCP and Usenet, O'Reilly & Associates, Inc. USA.

NAME

uucico - transfer files for the uucp system

SYNOPSIS

/usr/lbin/uucp/uucico -r1 -s *system* [**-x** *debug_level*] [**-d** *spool_directory*]

/usr/lbin/uucp/uucico [**-x** *debug_level*] [**-d** *spool_directory*]

DESCRIPTION

uucico scans the **/var/spool/uucp** directories for work files. If such files exist, a connection to a remote system is attempted using the line protocol for the remote system specified in file **/etc/uucp/Systems**. **uucico** then executes all requests for work and logs the results.

Options

uucico recognizes the following options:

-r1 Start **uucico** in the MASTER mode. The default is SLAVE mode.

-s *system* Do work only for the system specified by *system*. If there is no work for *system* on the local spool directory, initiate a connection to *system* to determine if *system* has work for the local system. This option must be used if **-r1** is specified.

-d *spool_directory*
Search directory *spool_directory* instead of the default spool directories (usually **/var/spool/uucp/***).

-x *debug_level* Use debugging option. *debug_level* is an integer in the range 1 through 9. More debugging information is given for larger values of *debug_level*.

uucico is usually started by a local program such as **cron**, **uucp**, or **uuxqt** (see *cron*(1M), *uucp*(1), and *uuxqt*(1M)). It when debugging should a user initiate **uucico** directly.

When started by a local program, **uucico** is considered the MASTER and attempts a connection to a remote system. If **uucico** is started by a remote system, it is considered to be in SLAVE mode.

For the **uucico** connection to a remote system to be successful, there must be an entry in the **/etc/passwd** file on the remote system of the form:

```
uucp::5:5::/var/spool/uucppublic:/usr/lbin/uucp/uucico
```

FILES

/etc/uucp/Systems
/etc/uucp/Permissions
/etc/uucp/Devices
/etc/uucp/Maxuuxqts
/etc/uucp/Maxuuscheds
/var/spool/uucp/*
/var/spool/locks/LCK*
/var/spool/uucppublic/*

SEE ALSO

uucp(1), uustat(1), uux(1), cron(1M), uusched(1M).

Tim O'Reilly and Grace Todino,
Managing UUCP and Usenet, O'Reilly & Associates, Inc. USA.

Grace Todino and Dale Dougherty,
Using UUCP and Usenet, O'Reilly & Associates, Inc. USA.

NAME

uuclean - uucp spool directory clean-up

SYNOPSIS

/usr/lbin/uucp/uuclean [*options*]

DESCRIPTION

uuclean scans the spool directories for files with the specified prefix and deletes all those that are older than the specified number of hours.

Options

uuclean recognizes the following options:

-d*directory*	Clean *directory* instead of the spool directory. If *directory* is not a valid spool directory, it cannot contain "work files"; i.e., files whose names start with **C.**. These files have special meaning to **uuclean** pertaining to **uucp** job statistics.
-p*pre*	Scan for files with *pre* as the file prefix. Up to 10 **-p** arguments can be specified. A **-p** without any *pre* following will cause all files older than the specified time to be deleted.
-n*time*	Files whose age is more than *time* hours are deleted if the prefix test is satisfied (default time is 72 hours).
-w*file*	The default action for **uuclean** is to remove files that are older than a specified time (see **-n** option). The **-w** option is used to find files older than *time* hours; however, the files are not deleted. If the argument *file* is present the warning is placed in *file*; otherwise, the warnings go to the standard output.
-s*sys*	Only files destined for system *sys* are examined. Up to 10 **-s** arguments can be specified.
-m*file*	The **-m** option sends mail to the owner of the file when it is deleted. If a *file* is specified, an entry is placed in *file*.

This program is typically started by **cron** (see *cron*(1M)).

FILES

/var/spool/uucp/* spool directory

SEE ALSO

uucp(1), uux(1), cron(1M), uucleanup(1M).

Tim O'Reilly and Grace Todino,
Managing UUCP and Usenet, O'Reilly & Associates, Inc. USA.

Grace Todino and Dale Dougherty,
Using UUCP and Usenet, O'Reilly & Associates, Inc. USA.

NAME

uucleanup - uucp spool directory clean-up

SYNOPSIS

/usr/lbin/uucp/uucleanup [**-C** *time*] [**-D** *time*] [**-W** *time*] [**-X** *time*] [**-m** *string*] [**-o** *time*] [**-s** *system*] [**-x** *debug_level*]

DESCRIPTION

uucleanup scans the spool directories for old files and takes appropriate action to remove them. Depending on the options selected, **uucleanup** performs the following:

- Informs the requestor of send and/or receive requests for systems that cannot be reached.
- Returns mail that cannot be delivered to the sender.
- Removes all other files.

In addition, **uucleanup** warns users of requestors who have been waiting for a given number of days (the default is 1 day). Note that unless *time* is specifically set, the default *time* values for the following options are used.

Options

uucleanup recognizes the following options:

-C*time*	Any **C.** files greater or equal to *time* days old are removed with appropriate information to the requestor. The default *time* is 7 days.
-D*time*	Any **D.** files greater or equal to *time* days old are removed. An attempt is made to deliver mail messages and execute news when appropriate. The default *time* is 7 days.
-W*time*	Any **C.** files equal to *time* cause a message to be mailed to the requestor warning about the delay in contacting the remote. The message includes the *JOBID,* and in the case of mail, the mail message. The administrator can include a message line telling who to call to correct the problem (see the **-m** option). The default *time* is 1 day.
-X*time*	Any **X.** files greater than or equal to *time* days old are removed. The **D.** files are probably not present (if they were, the **X.** could be executed). But, if **D.** files are present, they are taken care of by **D.** processing. The default *time* is 2 days.
-m*string*	This string is included in the warning message generated by the **-W** option. The default string is **See your local administrator to locate the problem.**
-o*time*	Other files whose age is more than *time* days are deleted. The default time is 2 days.
-s*system*	Clean-up the spool directory for *system* only. The default is to clean-up all spool directories.
-x*debug_level*	The debug level is a single digit between 0 and 9. The higher the numbers, the more detailed the debugging information returned.

This program is typically started by the script **uudemon.cleanu**, which should be started by **cron** (see *cron*(1M)).

FILES

/var/spool/uucp/* spool directory

SEE ALSO

cron(1M), uucp(1), uux(1), uuclean(1M).

Tim O'Reilly and Grace Todino,
Managing UUCP and Usenet, O'Reilly & Associates, Inc. USA.

Grace Todino and Dale Dougherty,
Using UUCP and Usenet, O'Reilly & Associates, Inc. USA.

NAME

uucpd - UUCP over TCP/IP server daemon

SYNOPSIS

`/usr/sbin/uucpd`

DESCRIPTION

uucpd is the server for supporting UUCP connections over TCP/IP networks.

uucpd is invoked by *inetd*(1M) when a UUCP connection is established (that is, a connection to the port indicated in the "uucp" service specification; see *services*(4)), and executes the following protocol:

1) The server prompts with "login:", the uucico process at the other end must supply a username.
2) Unless the username refers to an account without a password, the server then prompts with "Password:", the uucico process at the other end must supply the password for that account.

If the username is not valid or is valid but refers to an account that does not have **/usr/lbin/uucp/uucico** as its login shell, or if the password is not the correct password for that account, the connection is dropped. Otherwise, *uucico*(1M) is run. Entries are made in **/var/adm/wtmp** traceable with *who*(1) and *last*(1).

PROTOCOL RESTRICTION

Only 'g' protocol for uucico is supported.

DIAGNOSTICS

All diagnostic messages are returned on the connection, after which the connection is closed.

user read
An error occurred while reading the username.

passwd read
An error occurred while reading the password.

login incorrect
The username or the password is invalid or the user's login shell for this account is not /usr/lbin/uucp/uucico.

WARNINGS

On Trusted Systems uucpd prohibits uucico to start if any of the following are true :

- the login account is locked (several causes).
- current time doesn't match existing time-of-day restrictions for this account.

Under such conditions uucpd will return the message **login incorrect** to the connection. The connection is then dropped.

AUTHOR

uucpd was developed by the University of California, Berkeley and HP.

FILES

`/etc/inetd.conf`	configuration file for inetd
`/var/adm/inetd.sec`	optional security file for inetd
`/etc/services`	service name data base
`/var/adm/wtmp`	login data base

SEE ALSO

inetd(1M), services(4), uucico(1M).

NAME

uugetty - set terminal type, modes, speed and line discipline

SYNOPSIS

/usr/lbin/uucp/uugetty [**-h**] [**-t** *timeout*] [**-r**] *line* [*speed* [*type* [*linedisc*]]]

/usr/lbin/uucp/uugetty -c *file*

DESCRIPTION

uugetty sets terminal type, modes, speed and line discipline. It is similar to **getty**, except that **uugetty** supports using the line in both directions (see *getty*(1M)). This allows users to log in, but, if the line is free, **uucico**, **cu**, and **ct** can dial out (see *uucico*(1), *cu*(1), and *ct*(1)). When devices are used with **uucico**, **cu**, and **ct**, lock files are created. Therefore, when the call to **open()** returns (see *open*(2)) (or the first character is read when the **-r** option is used), the status of the lock files indicates whether the line is used by **uucico**, **cu**, **ct**, or someone trying to log in. See *getty*(1M) for more information.

Note that with the **-r** option, several carriage-return characters might be required before the login message is output. When **uucico** is trying to log in, it can be instructed to enter numerous carriage-return characters with the following login script:

```
\r\d\r\d\r\d\r in:-in: ...
```

where ... represents whatever would normally be used for the login sequence.

An entry for an intelligent modem or direct line that has a **uugetty** on each end must use the **-r** option (this causes *uugetty* to wait to read a character before it enters the login message, thus preventing two instances of **uugetty** from looping). If there is a **uugetty** on one end of a direct line, there must be a **uugetty** on the other end as well.

EXAMPLES

The following line is an **/etc/inittab** entry using **uugetty** on an intelligent modem or direct line:

```
30:2:respawn:/usr/lbin/uucp/uugetty -r -t 60 tty12 1200
```

WARNINGS

ct does not work when **uugetty** is used with an intelligent modem such as a Penril or a Ventel.

FILES

/etc/gettydefs
/etc/issue
/var/spool/locks/LCK*

SEE ALSO

ct(1), cu(1), login(1), uucico(1M), getty(1M), init(1M), ioctl(2), gettydefs(4), inittab(4), tty(7).

Tim O'Reilly and Grace Todino,
Managing UUCP and Usenet, O'Reilly & Associates, Inc. USA.

Grace Todino and Dale Dougherty,
Using UUCP and Usenet, O'Reilly & Associates, Inc. USA.

NAME

uuls - list spooled uucp transactions grouped by transaction

SYNOPSIS

uuls [**-m**] [*directories* ...]

uuls [**-s**] [**-m**] [*directories* ...]

uuls [**-k**] [**-m**] [*directories* ...]

DESCRIPTION

This command lists the contents of UUCP spool directories (default **/var/spool/uucp/***) with the files in each directory grouped into three categories:

- Transactions,
- Orphans, and
- Others.

Transactions

Each output line starts with a transaction control filename, and includes the name of each local (same-directory) subfile referenced by the control file (see below). Each is possibly followed by the total size in bytes (**-s** option) or Kbytes (**-k** option) in the transaction (see below). The **-m** (meanings) option replaces the subfile names with nodename, user, and *commandline* information (see below).

Orphans

All subfiles not referenced by any control file.

Others

All other files in the directory (all files not listed under one of the above categories).

Filenames are formatted into columns, so there can be more than one file per line. If a transaction has more subfiles than fit on one line, it is followed by continuation lines which are indented further.

The **-s** (size in bytes) and **-k** (Kbytes) options cause the command to follow each transaction in the **Transactions** section with a total size for all stat-able, sendable files in that transaction. This includes **D.*** files only, not **C.*** or **X.*** files. It does include stat-able files outside the spool directory that are indirectly referenced by **C.*** files. Sizes are either in bytes or rounded to the nearest Kbyte (1024 bytes), respectively. A totals line is also added at the end of the **Transactions** section.

The **-m** (meanings) option causes the command to follow **C.*** and **X.*** files with a *nodename***!***username commandline* line, instead of subfilenames. For **C** files, one line is printed per remote execution (**D***X***) subfile it references. *nodename* is truncated at seven characters, *username* at eight, and *commandline* at however much fits on one line.

If **-m** is given, for each **C** file with no remote execution files, the command instead shows the meaning of the **C** file itself on one or more lines. Each line consists of a username, then **R** (receive) or **S** (send), then the name of the file to be transferred. See below for details.

Filenames are listed in ascending collation order within each section (see Environment Variables below), except that the first section is only sorted by the control filename. Every file in the directory except **.** and **..** appears exactly once in the entire list, unless **-m** is used.

u

Details

Transaction files are those whose names start with **C.** or **X.**. Subfilenames, which usually start with **D.**, are gleaned from control file lines, at most one per line, from blank-separated fields, as follows:

C.*: R <remotefrom> <localto> <user> -<options>
C.*: S <localfrom> <remoteto> <user> -<options> <subfile> <mode>
X.*: F <subfile>

Lines that do not begin with the appropriate character (**R**, **S**, or **F**) are ignored.

In the **R** (receive) case, <remotefrom> is used to print the **C**-file meaning, and its transaction size is taken as zero (unknown).

In the **S** (send) case, if <subfile> is **D.0**, <localfrom> is a file not in the spool directory, resulting from a typical **uucp** call without the **-C** (copy) option. In this case <localfrom> is used for the transaction size, if stat-able, and to print the **C**-file meaning.

uucp -C and **uux** both set <subfile> to a true (spooled) subfile name.

Orphan files are those whose names start with **D.** and which are not referenced by any control files.

This algorithm extracts from control files the names of all subfiles that should exist in the spool directory when the transaction is not being actively processed. It is not unusual to see "missing subfiles" and "orphans" if you **uuls** a spool directory while **uucico**, **uucp**, **uux**, or **uuxqt** is active.

Meanings information is obtained by reading each **D**∗**X**∗ subfile referenced by each **C.**∗ file, and by reading **X**∗**X**∗ files. *nodename***!***username* is taken from the last line in the file which is of the form:

U <username> <nodename>

Likewise, *commandline* is taken from the last line of the form:

C <commandline>

If a subfile name is referenced more than once, references after the first show the subfile as missing. If a subfile name appears in a (corrupt) directory more than once, the name is only found once, but then it is listed again under **Orphans**.

EXTERNAL INFLUENCES

Environment Variables

LC_COLLATE determines the order in which the output is sorted.

If LC_COLLATE is not specified in the environment or is set to the empty string, the value of LANG is used as a default. If LANG is not specified or is set to the empty string, a default of “C” (see *lang*(5)) is used instead of LANG. If any internationalization variable contains an invalid setting, **uuls** behaves as if all internationalization variables are set to “C” (see *environ*(5)).

DIAGNOSTICS

The program writes an appropriate message to standard error if it has any problems dealing with a specified file (directory), including failure to get heap space. It always returns zero as its exit value.

If a control file is unopenable (wrong permissions or it disappeared while **uuls** was running), its name is preceded by a ∗ and the size of the transaction is zero. If a subfile is missing (filename not found in the directory being listed) or not stat-able (if required for **-s** or **-k**), its name is preceded by a ∗ and it contributes zero bytes to the size of the transaction.

If **-m** is specified and a **D**∗**X**∗ file is missing or unreadable, its name is given with a ∗ prefixed, as usual.

BUGS

This command uses *chdir*(2) to change to each directory in turn. If more than one is specified, the second through last directories must be absolute (not relative) pathnames, or the *chdir*() may fail.

AUTHOR

uuls was developed by HP.

SEE ALSO

mail(1), uucp(1), uuto(1), uux(1), uuxqt(1M), stat(2).

NAME

uusched - schedule uucp transport files

SYNOPSIS

/usr/lbin/uucp/uusched [**-u** *debug_level*] [**-x** *debug_level*]

DESCRIPTION

uusched is the UUCP file transport scheduler. It is usually started by the daemon **uudemon.hour**, which is started by **cron** (see *cron*(1M)) from the following entry in **/var/spool/cron**:

```
39 * * * * /usr/bin/su uucp -c */usr/lbin/uucp/uudemon.hour > /dev/null*
```

Options

uusched recognizes two options which are provided for debugging purposes only.

-x *debug_level*	Output debugging messages.
-u *debug_level*	Pass as **-x** to **uucico** (see *uucico*(1M)). The *debug_level* is a number between 0 and 9. The higher the number, the more detailed the information returned.

FILES

/etc/uucp/Systems
/etc/uucp/Permissions
/etc/uucp/Devices
/var/spool/uucp/*
/var/spool/locks/LCK*
/var/spool/uucppublic/*

SEE ALSO

cron(1M), uucico(1M), uusched(1M), uucp(1), uustat(1), uux(1).

Tim O'Reilly and Grace Todino,
Managing UUCP and Usenet, O'Reilly & Associates, Inc. USA.

Grace Todino and Dale Dougherty,
Using UUCP and Usenet, O'Reilly & Associates, Inc. USA.

NAME

uusnap - show snapshot of the UUCP system

SYNOPSIS

uusnap

DESCRIPTION

uusnap displays in tabular format a synopsis of the current UUCP situation. The format of each line is as follows:

site *N Cmds* *N Data* *N Xqts* *Message*

Where *site* is the name of the site with work, *N* is a count of each of the three possible types of work (command, data, or remote execute), and *Message* is the current status message for that site as found in the **STST** file.

Included in *Message* may be the time left before UUCP can re-try the call, and the count of the number of times that UUCP has tried to reach the site. The process id of **uucico** may also be shown if it is in a TALKING state.

AUTHOR

uusnap was developed by the University of California, Berkeley.

SEE ALSO

uucp(1).

Tim O'Reilly and Grace Todino,
Managing UUCP and Usenet, O'Reilly & Associates, Inc. USA.

Grace Todino and Dale Dougherty,
Using UUCP and Usenet, O'Reilly & Associates, Inc. USA.

NAME

uusnaps - sort and embellish uusnap output

SYNOPSIS

uusnaps

DESCRIPTION

uusnaps runs **uusnap** (see *uusnap*(1M)) and post-processes the output into a more useful form. It sorts output lines in "Pareto-style", showing first those remote systems with the greatest number of **Cmds** files, next **Data** files, and then **Xqts** files.

uusnaps inserts a ***** after the number of **Xqts** files on those lines where **Data** is not equal to (2 × **Cmds**) + **Xqts**. This may be a sign of missing or orphaned transaction parts. Use **uuls** to check (see *uuls*(1)).

uusnaps adds summary information after all **uusnap** output. The first line is a total of the numbers of **Cmds**, **Data**, and **Xqts** files. The second line contains a grand total number of transaction files, followed by the number of directory bytes this represents. This is an indication of the true size of the directory itself if all empty entries were squeezed out. Finally, if it appears that transaction files might be missing or orphaned, **uusnaps** returns the number of missing or excess files.

WARNINGS

uusnaps assumes that each directory entry takes 24 bytes.

SEE ALSO

uusnap(1M), uuls(1).

NAME

uusub - monitor uucp network

SYNOPSIS

/usr/lbin/uucp/uusub [*options*]

DESCRIPTION

uusub defines a **uucp** subnetwork and monitors the connection and traffic among the members of the subnetwork.

Options

uusub recognizes the following options:

-a*sys*	Add *sys* to the subnetwork.
-d*sys*	Delete *sys* from the subnetwork.
-l	Report the statistics on connections.
-r	Report the statistics on traffic amount.
-f	Flush the connection statistics.
-u*hr*	Gather the traffic statistics over the past *hr* hours.
-c*sys*	Exercise the connection to the system *sys*. If *sys* is specified as **all**, exercise the connection to all the systems in the subnetwork.

The connections report is formatted as follows:

sys #call #ok time #dev #login #nack #other

Format interpretation:

sys	remote system name,
#call	number of times the local system tried to call *sys* since the last flush was done,
#ok	number of successful connections,
time	latest successful connect time,
#dev	number of unsuccessful connections because of no available device (e.g., ACU),
#login	number of unsuccessful connections because of login failure,
#nack	number of unsuccessful connections because of no response (e.g. line busy, system down),
#other	number of unsuccessful connections because of other reasons.

Traffic statistics are reported as follows:

sfile sbyte rfile rbyte

Format interpretation:

sfile	number of files sent,
sbyte	number of bytes sent over the period of time indicated in the latest *uusub* command with the **-u***hr* option,
rfile	number of files received,
rbyte	number of bytes received.

The command:

```
uusub -c all -u 24
```

is typically started by **cron** once a day.

FILES

/var/uucp/.Admin/L_sub	connection statistics
/var/uucp/.Admin/R_sub	traffic statistics

`/var/uucp/.Log/*`	system log file

SEE ALSO

uucp(1), uustat(1).

Tim O'Reilly and Grace Todino,
Managing UUCP and Usenet, O'Reilly & Associates, Inc. USA.

Grace Todino and Dale Dougherty,
Using UUCP and Usenet, O'Reilly & Associates, Inc. USA.

NAME

uuxqt - execute remote uucp or uux command requests

SYNOPSIS

/usr/lbin/uucp/uuxqt [**-s** *system*] [**-x** *debug_level*]

DESCRIPTION

uuxqt executes remote job requests generated by use of the **uux** command (see *uux*(1)). **uux** generates **X.** files and places them in the spool directory, where **uuxqt** searches for them. For each **X.** file, **uuxqt** determines whether the required data files are available and accessible, and if file commands are permitted for the requesting system. The **Permissions** file is used to validate file accessibility and command execute permission. Then **uuxqt** performs execution of the commands.

Two environment variables are set before the **uuxqt** command is executed: **UU_MACHINE** is the machine that sent the previous job and **UU_USER** is the user who sent the job. These can be used in writing commands that remote systems can execute to provide information, auditing, or restrictions.

uuxqt recognizes the following options:

-s *system*	Execute commands on the specified *system* .
-x *debug_level*	Produce debugging output on standard output. *debug_level* is a single digit between 0 and 9. The higher the number, the more detailed debugging information returned.

FILES

/etc/uucp/Permissions
/etc/uucp/Maxuuxqts
/var/spool/uucp/*
/var/spool/locks/LCK*

SEE ALSO

uucp(1), uustat(1), uux(1), uucico(1M).

Tim O'Reilly and Grace Todino,
Managing UUCP and Usenet, O'Reilly & Associates, Inc. USA.

Grace Todino and Dale Dougherty,
Using UUCP and Usenet, O'Reilly & Associates, Inc. USA.

NAME

vgcfgbackup - create or update LVM volume group configuration backup file

SYNOPSIS

/usr/sbin/vgcfgbackup [**-f** *vg_conf_path*] [**-u**] *vg_name*

DESCRIPTION

The **vgcfgbackup** command saves the LVM configuration for a volume group in a default or alternate configuration backup file (see the **-f** option).

By default, **vgcfgbackup** runs automatically each time an LVM command changes the LVM configuration. In this case, it always uses the default configuration backup file. An existing default configuration backup file is renamed with an extension of **.old**.

Options and Arguments

vgcfgbackup recognizes the following options and arguments:

vg_name	The path name of a volume group.
-f *vg_conf_path*	Save the configuration using an alternate file name specified by *vg_conf_path*. If **-f** is omitted, the default file name is in the form: **/etc/lvmconf/***base_vg_name***.conf** *base_vg_name* is the base name of *vg_name*. For example, if *vg_name* is specified as **/dev/vg00**, *base_vg_name* is **vg00**.
-u	Update the configuration backup file with the latest LVM configuration. Only those physical volumes added since the configuration backup file was last modified need to be online. If **-u** is omitted, all physical volumes for *vg_name* must be online.

RETURN VALUE

vgcfgbackup exits with one of the following values:

0	Successful completion.
>0	Failure. Errors occurred when information from the volume group was being accessed.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages are displayed.

If **LANG** is not specified or is null, it defaults to "C" (see *lang*(5)).

If any internationalization variable contains an invalid setting, all internationalization variables default to "C" (see *environ*(5)).

EXAMPLES

Back up LVM configuration information for volume group **/dev/vg00** in the default backup file **/etc/lvmconf/vg00.conf**:

```
vgcfgbackup /dev/vg00
```

Update LVM configuration information corresponding to volume group **/dev/vg00** in the default backup file **/etc/lvmconf/vg00.conf**:

```
vgcfgbackup -u /dev/vg00
```

Back up LVM configuration information for volume group **/dev/vg00** in the alternate configuration backup file **/tmp/vg00.backup**:

```
vgcfgbackup -f /tmp/vg00.backup vg00
```

WARNINGS

It is recommended that any alternate configuration backup file be created in the root file system (as is the case with the default path name). This facilitates easy volume group recovery during maintenance mode, such as after a system crash.

AUTHOR

`vgcfgbackup` was developed by HP.

SEE ALSO

vgcfgrestore(1M).

NAME

vgcfgrestore - display or restore LVM volume group configuration from backup file

SYNOPSIS

/usr/sbin/vgcfgrestore -n *vg_name* **-l**

/usr/sbin/vgcfgrestore [-R] [-F] -n *vg_name* [**-o** *old_pv_path*] *pv_path*

/usr/sbin/vgcfgrestore -f *vg_conf_path* **-l**

/usr/sbin/vgcfgrestore [-R] [-F] -f *vg_conf_path* [**-o** *old_pv_path*] *pv_path*

Remarks

vgcfgrestore cannot be performed if the volume group is activated in shared mode.

DESCRIPTION

The **vgcfgrestore** command restores the LVM configuration data from a default (**-n** option) or alternate (**-f** option) configuration backup file to the physical volume named by *pv_path*. Or, it displays the configuration data on standard output (**-l** option).

The configuration stored for one physical volume, *old_pv_path*, can be copied to another physical volume *pv_path* (**-o** option).

Options and Arguments

vgcfgrestore recognizes the following options and arguments:

pv_path — The raw (character) device path name of a physical volume that is currently online.

If the **-o** option is omitted, *pv_path* must specify a physical volume whose configuration is stored in the configuration backup file.

-f *vg_conf_path* — Get configuration information from the alternate configuration backup file *vg_conf_path*.

-F — This option will force restoring the LVM configuration data even if the physical volume has alternate block(s) allocated inside the user data area. This option should be used with extreme caution. User is advised to fix the problem because potential data corruption could occur.

-l — List the configuration information saved in the specified configuration backup file.

-n *vg_name* — Get configuration information from the default configuration backup file:

/etc/lvmconf/*base_vg_name***.conf**

vg_name is the path name of the volume group.

base_vg_name is the base name of *vg_name*. For example, if *vg_name* is specified as **/dev/vg00**, *base_vg_name* is **vg00**.

-o *old_pv_path* — Restore the configuration information saved for physical volume *old_pv_path* to physical volume *pv_path*.

This option is useful when a physical volume's name has changed since the configuration backup file was created or updated.

old_pv_path must be the path name of a physical volume whose configuration is stored in the configuration backup file. It need not be currently online.

pv_path must be the path name of a physical volume that is currently online. Its configuration need not be stored in the configuration backup file.

-R — This option will force restoring the LVM configuration data even if there is a physical volume mismatch between the kernel and the configuration backup file with the volume group still active. This option should not be used unless the configuration file is absolutely valid and up-to-date. Restoring invalid configuration data can result in data corruption later.

If there are alternate physical volume links configured in the system, the following message will appear when total number of physical volumes in the kernel does not match with the configuration backup file due to missing alternate physical volume links:

```
Mismatch between the backup file and the running kernel:
Kernel indicates X disks for /dev/vgname; /etc/lvmconf/vgname
indicates Y disks.  Cannot proceed with the restoration.
Deactivate the Volume Group and try again.
```

In this case, the user is advised to deactivate the volume group first, then use the **vgcfgrestore** command to restore configuration data when the volume group is unavailable. But if the volume group has to stay available and the user is absolutely sure the configuration file is correct, this option will restore data from the configuration file when the volume group stays available.

RETURN VALUE

vgcfgrestore exits with one of the following values:

0	Successful completion.
>0	Failure. Errors occurred during the restore operation.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages are displayed.

If **LANG** is not specified or is null, it defaults to "C" (see *lang*(5)).

If any internationalization variable contains an invalid setting, all internationalization variables default to "C" (see *environ*(5)).

EXAMPLES

Restore the LVM configuration information for the physical volume **/dev/rdsk/c0t7d0** that was saved in the default file **/etc/lvmconf/vg00.conf**:

```
vgcfgrestore -n /dev/vg00 /dev/rdsk/c0t7d0
```

Force to restore the LVM configuration data when volume group is still active

```
vgcfgrestore -R -n /dev/vg00 /dev/rdsk/c0t7d0
```

Restore the LVM configuration information to physical volume **/dev/rdsk/c0t4d0** using alternate configuration file **/tmp/vg00.backup**:

```
vgcfgrestore -f /tmp/vg00.backup /dev/rdsk/c0t4d0
```

List backup information saved in default configuration file **/etc/lvmconf/vg00.conf**:

```
vgcfgrestore -n /dev/vg00 -l
```

Above command might display the following:

```
Volume Group Configuration information in "/etc/lvmconf/vg00.conf"
VG Name /dev/vg00
 ---- Physical volumes : 2 ----
    /dev/rdsk/c0t6d0 (Bootable)
    /dev/rdsk/c0t5d0 (Non-bootable)
```

Restore LVM configuration information stored for **/dev/rdsk/c0t7d0** in default configuration file **/etc/lvmconf/vg01.conf** to physical volume **/dev/rdsk/c0t6d0**:

```
vgcfgrestore -n /dev/vg01 -o /dev/rdsk/c0t7d0 /dev/rdsk/c0t6d0
```

WARNINGS

Preferably, the volume group should be made unavailable before executing **vgcfgrestore** by executing the command

vgchange -a n *vg_name*

AUTHOR

vgcfgrestore was developed by HP.

SEE ALSO

vgcfgbackup(1M).

NAME

vgchange - set LVM volume group availability

SYNOPSIS

Activate volume group

/usr/sbin/vgchange -a *availability* [**-l**] [**-p**] [**-q** *quorum*] [**-s**] [**-P** *resync_daemon_count*] [*vg_name* ...]

Assign to high availability cluster and mark volume group sharable

/usr/sbin/vgchange -c *cluster* **-S** *sharable vg_name*

Remarks

MC/ServiceGuard cluster operations require the installation of the optional MC/ServiceGuard software, which is not included in the standard HP-UX operating system.

Lock Manager cluster operations require the installation MC/LockManager software which is not included with the standard HP-UX operating system.

Mirrored disk operations require the installation of the optional HP MirrorDisk/UX software, which is not included in the standard HP-UX operating system.

DESCRIPTION

The **vgchange** command with the **-a** option activates or deactivates one or more volume groups.

The **vgchange** command with the **-c** option controls the membership of one or more volume groups in a high availability cluster. The **vgchange** command with the **-c** and **-S** options control the membership of a volume group and mark it sharable.

The **vgchange** command without the **-P** *resync_daemon_count* option (default) will spawn one *nomwcsyncd* process for each **NOMWC/NONE** volume group being activated. This may create a lot of *nomwcsyncd* processes running concurrently when it activates a large number of **NOMWC/NONE** volume groups and overload.

The **-P** *resync_daemon_count* option provides a way to control the number of concurrent *nomwcsyncd* processes. The count is an advisory number and a different count might be chosen internally if load balance or other reason is needed. When specified, there are up to *resync_daemon_count* **+ 1** *nomwcsyncd* processes; one of them is the controlling processing to spawn others. **-P 0** will use the system default (currently defined to be 4).

vg_name must be defined as a volume group in the file **/etc/lvmtab**. If *vg_name* is omitted, all volume groups defined in **/etc/lvmtab** are affected.

High Availability Cluster Overview

Volume groups can be defined on disk volumes that are connected to two or more systems in a high availability cluster. This situation has a high potential for data corruption unless special software is used to coordinate shared access to the same volume group by all systems. This coordination is provided by MC/ServiceGuard or MC/LockManager.

A volume group can be marked as part of a MC/ServiceGuard cluster. When such a group is activated in exclusive mode, it can be accessed for exclusive read-write activity by only one of the systems at a time; the other systems can have read-only access to the data.

A volume group can be marked as a part of an MC/LockManager cluster. In this case, the volume group can be marked as sharable, and may be activated in shared mode for read-write access by all the nodes in the cluster. Shared read-write access by multiple cluster nodes is coordinated by MC/LockManager's distributed lock manager (DLM).

Options and Arguments

vgchange recognizes the following options and arguments:

vg_name The path name of a volume group.

-a *availability* Set volume group availability. *availability* can have one of the following values:

y Activate each specified volume group and all associated physical and logical volumes for read-write access. If a volume group is marked as part of a high availability cluster, it is activated in exclusive read-write mode, as for the **-a e** option.

V

- **e** Activate each specified volume group and all associated physical and logical volumes for exclusive read-write access. The volume group must be marked as part of a high availability cluster, and the availability software must be running on the system; otherwise, the volume group is not activated.

- **s** Activate each specified volume group and all associated physical and logical volume for shared read-write access. The volume group must be marked as part of a high availability cluster and marked sharable; otherwise, the volume group is not activated.

  If any of the logical volumes in the volume group are mirrored, and if there are more than two systems in the high availability cluster, this volume group will not be activated because HP MirrorDisk/UX software is only supported in a clustered environment with a maximum of two nodes configured.

If the **-a y** or **-a e** option is executed on a currently active volume group, **vgchange** attempts to include any physical volumes that were previously listed as missing. This is useful if a physical volume has come back online. However, no automatic synchronization of any mirrored logical volumes is done. If synchronization is required, execute the **vgsync** command (see *vgsync*(1M)).

- **r** Activate each specified volume group and all associated physical and logical volumes for read-only access. This option is ignored for a volume group that is already activated.

  If a volume group is marked as part of a high availability cluster, the high availability software must be running on the system; otherwise, the volume group is not activated.

- **n** Deactivate each specified volume group and its associated logical volumes. You must close the logical volumes prior to executing this option. For example, if the logical volume contains a file system, the file system must be unmounted.

-c *cluster* Control the membership of volume groups in a high availability cluster. *cluster* can have one of the following values:

- **y** Mark each specified volume group as a member of the high availability cluster. The high availability software must be running; otherwise, the volume group is not marked. Needs to be done on one node only.

- **n** Remove each specified volume group from membership in the high availability cluster. The high availability software does not need to be running.

The volume group must be deactivated with the **-a n** option before a **-c y|n** option can be executed.

-S *sharable* Control the sharability of volume groups in a high availability cluster. *sharable* can have one of the following values:

- **y** Mark each specified volume group as sharable. The high availability software must be running; otherwise, the volume group is not marked. Needs to be done on one node only.

- **n** Remove the shared attribute from the volume group. The high availability software does not need to be running.

The volume group must be deactivated with the **-a n** option before a **-S y|n** option can be executed.

-l Disable the opening of logical volumes that belong to each specified volume group. If the **-l** option is set, later attempts to open the logical volumes will fail. To allow an opening of these logical volumes to succeed, execute **lvchange -a y**.

-p Activate each specified volume group only if all of the physical volumes that belong to it are available.

-q *quorum* Set the quorum enforcement for each specified volume group. *quorum* can have one of the following values:

- **y** Enforce the quorum requirement. This is the default.
- **n** Ignore the quorum requirement.

The **-q n** option can be used to activate the volume group when the disk quorum is not maintained because too many disks were lost. Since it ensures the integrity of the LVM configuration information, it is normally not advisable to override the quorum.

-s Disable the synchronization of stale physical extents within the volume group specified by *vg_name*. This option is only effective when used with the **-a y** or **-a e** option.

-P *resync_daemon_count*
gives the advisory count to control the number of *nomwcsyncd* processes on volume group activation.

Mirrored Disk Activation

When the optional HP MirrorDisk/UX software is running and a volume group is activated, LVM performs the necessary mirror consistency recovery for each logical volume in the volume group based on the state of Mirror Write Cache and Mirror Consistency Recovery (see the Consistency Recovery section of *lvdisplay*(1M)). In a non-shared environment, LVM supports **MWC**, **NOMWC** and the **NONE** recovery. But in shared environment, LVM only supports **NOMWC** and the **NONE** recovery.

MWC Recover mirror consistency by using the Mirror Write Cache and Mirror Consistency Record. This mode implies that the Mirror Write Cache is on.

NOMWC Recover mirror consistency by searching all logical extents and copying data from a non-stale copy to the other mirror copies. This mode implies that the Mirror Write Cache is off.

NONE Do not recover mirror consistency during volume group activation on this logical volume. This mode implies that the Mirror Write Cache is off.

Next, mirror synchronization refreshes stale mirror copies by copying data from a nonstale copy. If the **-s** option is specified on the command line, mirror synchronization does not occur. However, for those logical volumes that have Mirror Write Cache turned off, mirror synchronization is done independently of whether the **-s** option appears on the command line.

General Activation

If **vgchange** cannot access a physical volume, it lists the volume's status as missing. If too many physical volumes in the volume group are missing, **vgchange** reports that the group does not have a quorum and cannot be activated. The lack of a quorum can be overridden with the **-q n** option.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages are displayed.

If **LANG** is not specified or is null, it defaults to "C" (see *lang*(5)).

If any internationalization variable contains an invalid setting, all internationalization variables default to "C" (see *environ*(5)).

EXAMPLES

Activate volume group **/dev/vg03**:

```
vgchange -a y /dev/vg03
```

Deactivate volume group **/dev/vg03**:

```
vgchange -a n /dev/vg03
```

Activate volume group **/dev/vg03** without synchronizing extents that are not current on logical volumes that have Mirror Write Cache turned on:

```
    vgchange -a y -s /dev/vg03
```

Exclusive Activation

Set up volume group **/dev/vg03** for use in a high availability cluster:

```
    vgchange -a n /dev/vg03   # Deactivate volume group
    vgchange -c y /dev/vg03   # Enable volume group for HA cluster
    vgchange -c y -S y /dev/vg03 # Enable volume group for HA cluster
                                   and mark as sharable
    vgchange -a e /dev/vg03   # Activate volume group in exclusive mode
    vgchange -a s /dev/vg03   # Activate volume group in shared mode
```

Activate all volume groups; activate those that are marked for membership in a high availability cluster in exclusive mode:

```
    vgchange -a y
```

Activate all volumes that are marked for membership in a high availability cluster in exclusive mode:

```
    vgchange -a e
```

WARNINGS

Ordinary Operation

In ordinary operation (i.e., without the optional high availability software), it is possible to activate a volume group for read-write access from more than one physically connected system, leading to a high potential for data corruption. Therefore, if access is desired from more than one system to a single volume group, it is important that only one system activate the volume group for read-write access; the other systems can use read-only access. There is no problem if all systems activate the volume group for read-only access.

Furthermore, volume group information is only read from the disks during volume group activation. Dynamic changes to the volume group such as the following are not propagated to other systems sharing the volume group:

Logical volume configuration changes.

Changes to the status of the mirrored extents.

Bad-block relocation that occurs during write operations.

Because of these limitations, when sharing volume groups between systems it is recommended that logical volumes be accessed only by one system at a time. If logical volumes need to be accessed simultaneously, the logical volumes should not be mirrored and should not have bad-block relocation turned on, or all systems should use read-only access to the logical volumes.

SEE ALSO

mount(1M), vgcreate(1M), vgextend(1M), vgreduce(1M), vgdisplay(1M).

If MC/ServiceGuard is installed: cmcheckconf(1M), cmquerycl(1M), and *Managing MC/ServiceGuard*.

NAME

vgchgid - modify the Volume Group ID (VGID) on a given set of physical devices

SYNOPSIS

/usr/sbin/vgchgid *PhysicalVolumePath* [*PhysicalVolumePath*] ...

DESCRIPTION

The **vgchgid** command is designed to change the LVM Volume Group ID (VGID) on a supplied set of disks. It is primarily targeted for disk arrays which have capability of creating mirrored Business Copy (BC), such as EMC Symmetrix disks, and the XP disk array family, such as XP256 and XP512. **vgchgid** command accepts a set of raw physical devices and checks the following criteria before it alters the VGID:

- All raw physical volume devices in the command line have the same disk type, such as:

  1) EMC Symmetrix disks with the BCV attributes. (See EMC documentation.)
  2) The XP disk array family with BC_SVOL or CA_SVOL attributes. (See XP256/XP512 related documentation.)

- All raw physical volume devices in the command line belong to the same VG. (See WARNINGS section.)

Once the checks are successful, the same VGID is set on all the disks. It should be noted that for multi-PV volume groups all the physical volumes should be split-off and supplied in a single invocation of the **vgchgid** command.

Options

vgchgid recognizes the following options and arguments:

PhysicalVolumePath The raw devices path name of a physical volume.

Background

Both the EMC and XP disk arrays have a feature which allows a user to split-off a set of mirrored copies of physical volumes (termed **BCV**s or **BC**s) just as LVM split-off logical volumes with **lvsplit** command. As the result of the "split," the split-off devices will have the same VGID as the original disks. The **vgchgid** command is needed to modify the VGID on the BCV devices. Once the VGID has been altered, the BCV disks can be imported into a new volume group by using the **vgimport** command.

WARNINGS

Once the VGID has been changed, the original VGID is lost until a BCV device is re-mirrored with the original devices. If the **vgchgid** command is used on a subset of BCV devices (e.g., two out of four BCV devices), the two groups of BCV devices would not be able to be imported into the same VG since they have different VGID on them. The solution is to re-mirror all four of the BCV devices and re-run the **vgchgid** command on all four BCV devices at the same time, and then use the **vgimport** command to import them into the same new VG.

If a disk is newly added to an existing volume group and no subsequent LVM operations has been performed to alter the structures (i.e., operations which perform an automated *vgcfgbackup*(1M)); then it is possible a subsequent **vgchgid** will fail. It will report that the disk does not belong to the volume group. This may be overcome by performing a structure changing operation on the volume group (for example, using **lvcreate**).

RETURN VALUE

vgchgid command returns the following values:

0	VGID was modified with no error
1	VGID was not modified

EXAMPLES

An example showing how **vgchgid** command might be used.

1. The system administrator uses the following commands to create the Business Continuity (BCV or BC) copy:

   1) For EMC Symmetrix disks, the commands are **BCV establish** and **BCV split**.

   2) For XP disk array, the commands are **paircreate** and **pairsplit**.

   Three BCV disks are created.

2. Change the VGID on the BCV disks.

   ```
   vgchgid /dev/rdsk/c0t0d0 /dev/rdsk/c0t0d1 /dev/rdsk/c0t0d2
   ```

3. Make a new volume group using the BCV disks.

   ```
   mkdir /dev/vgbcv
   mknod /dev/vgbcv/group c 64 0x040000
   ```

4. Import the BCV disks into the new volume group.

   ```
   vgimport /dev/vgbcv /dev/dsk/c0t0d0 /dev/dsk/c0t0d1 /dev/dsk/c0t0d2
   ```

5. Activate the new volume group.

   ```
   vgchange -a y /dev/vgbcv
   ```

6. Backup the new volume group's LVM data structure.

   ```
   vgcfgbackup /dev/vgbcv
   ```

7. Mount the associated logical volumes.

   ```
   mkdir /bcv/lvol1 /bcv/lvol2
   mount /dev/vgbcv/lvol1 /bcv/lvol1
   mount /dev/vgbcv/lvol2 /bcv/lvol2
   ```

SEE ALSO

vgimport(1M), vgscan(1M), vgcfgbackup(1M).

NAME

vgcreate - create LVM volume group

SYNOPSIS

/usr/sbin/vgcreate [**-f**] [**-A** *autobackup*] [**-x** *extensibility*] [**-e** *max_pe*] [**-l** *max_lv*]
[**-p** *max_pv*] [**-s** *pe_size*] [**-g** *pvg_name*] *vg_name pv_path* ...

DESCRIPTION

The **vgcreate** command creates a new volume group. *vg_name* is a symbolic name for the volume group and must be used in all references to it. *vg_name* is the path to a directory entry under **/dev** which must contain a character special file named **group**. Except for the **group** entry, the directory *vg_name* should be empty. The *vg_name* directory and the **group** file have to be created by the user (see *lvm*(7)).

vgcreate leaves the volume group in an active state.

Before assigning a physical volume to a volume group, the physical volume has to be created using the **pvcreate** command (see *pvcreate*(1M)).

If **vgcreate** fails to install the first specified physical volume into the volume group, the volume group is not created. If, for any reason, one of the remaining specified physical volumes cannot be installed into the volume group, an error message is printed, but the installation continues until the end of the list of physical volumes.

Options and Arguments

vgcreate recognizes the following options and arguments:

pv_path — The block device path name of a physical volume that will be assigned to the new volume group. You can specify physical volume links *(pv-links)* for a physical volume providing different paths that reference the same physical volume in the *pv_path* list. The order in which the paths are listed is important. The first path becomes the **primary link** to the physical volume, the second becomes an **alternate link** to the physical volume. The **primary link** is the default path used to access the physical volume. If the **primary link** becomes unavailable, LVM automatically switches to the **alternate link** to access the physical volume. Currently LVM supports a maximum of 8 paths to a physical volume (7 alternate and one primary).

vg_name — The path name of a subdirectory of the **/dev** directory. *vg_name* must be empty except for a character special file named **group**. Typically, this directory name is in the form **/dev/vg***NN*, where *NN* numbers sequentially from **00**.

-A *autobackup* — Set automatic backup for this invocation of this command. *autobackup* can have one of the following values:

- **y** Automatically back up configuration changes made to the volume group. This is the default.

  After this command executes, the **vgcfgbackup** command (see *vgcfgbackup*(1M)) is executed for the volume group.

- **n** Do not back up configuration changes this time.

-e *max_pe* — Set the maximum number of physical extents that can be allocated from any of the physical volumes in the volume group. The default value for *max_pe* is **1016**. However, if the size of any physical volume exceeds **1016** times the *pe_size*, the default value for *max_pe* is adjusted to match the physical volume size. The maximum number of physical extents can be a value in the range 1 to 65535.

-f — This option will force a volume group to be created with a physical volume which has alternate block(s) already allocated, (i.e. this physical volume was not initialized using **pvcreate -f**.) This option should be used with extreme caution. If the volume group to be created has a different physical extent size, the alternate block(s) might be inside the user data area. Potential data corruption could occur.

-g *pvg_name* — Create a new physical volume group with the name *pvg_name*. All physical volumes specified in the *pv_path* parameter become a member of the newly created physical volume group.

The physical volume group information is stored in an ASCII file, **/etc/lvmpvg**. The file can be edited to create a physical volume group instead of using the **vgcreate** command. However, ensure that the physical volumes to be used have already been installed in the volume group prior to creating the physical volume group.

The physical volume group name must be unique within a volume group although identical physical volume group names can appear in different volume groups (see *lvmpvg*(4) for format details).

-l *max_lv* Set the maximum number of logical volumes that the volume group is allowed to contain. The default value for *max_lv* is **255**. The maximum number of logical volumes can be a value in the range 1 to 255.

-p *max_pv* Set the maximum number of physical volumes that the volume group is allowed to contain. The default value for *max_pv* is **16**. The maximum number of physical volumes can be a value in the range 1 to 255.

-s *pe_size* Sets the number of megabytes in each physical extent, where *pe_size* is expressed in units of megabytes (MB) in the range 1 to 256. *pe_size* must be equal to a power of 2 (1, 2, 4, 8, etc.). The default value for *pe_size* is **4** (four megabytes).

-x *extensibility* Set the allocation permission for adding physical extents on the physical volumes specified by the *pv_path* parameter. *extensibility* can have one of the following values:

- **y** Allow allocation of additional physical extents on the physical volume. This is the default.
- **n** Prohibit allocation of additional physical extents on the physical volume. Logical volumes residing on the physical volume can still be accessed after the volume group has been activated by the **vgchange -a y** command.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages are displayed.

If **LANG** is not specified or is null, it defaults to "C" (see *lang*(5)).

If any internationalization variable contains an invalid setting, all internationalization variables default to "C" (see *environ*(5)).

EXAMPLES

Create a volume group named **/dev/vg00** containing two physical volumes with extent size set to 2 MB, from scratch.

First, create the directory **/dev/vg00** with the character special file called **group**.

```
mkdir /dev/vg00
mknod /dev/vg00/group c 64 0x030000
```

The minor number for the **group** file should be unique among all the volume groups on the system. It has the format **0x***NN***0000**, where *NN* runs from **00** to **ff**. The maximum value of *NN* is controlled by the kernel tunable parameter **maxvgs**.

Initialize the disks using *pvcreate*(1M).

```
pvcreate /dev/rdsk/c1t0d0
pvcreate /dev/rdsk/c1t2d0
```

Create the volume group.

```
vgcreate -s 2 /dev/vg00 /dev/dsk/c1t0d0 /dev/dsk/c1t2d0
```

Create a volume group named **/dev/vg01** that can contain a maximum of three logical volumes, with extent size set to 8 MB:

```
vgcreate -l 3 -s 8 /dev/vg01 /dev/dsk/c3t4d0
```

Create a volume group named **/dev/vg00** and a physical volume group named **PVG0** with two physical volumes:

```
vgcreate -g PVG0 /dev/vg00 /dev/dsk/c1t0d0 /dev/dsk/c2t0d0
```

Using the **PV**Links feature to create a volume group named **/dev/vg00** with a physical volume which can be referenced by two different paths. **/dev/dsk/c3t0d0** and **/dev/dsk/c4t0d0** refer to the same physical volume, accessed via different controller hardware paths. In this example, **/dev/dsk/c3t0d0** becomes the **primary link** to the physical volume. **/dev/dsk/c4t0d0** becomes an **alternate link** to the physical volume.

```
vgcreate /dev/vg00 /dev/dsk/c3t0d0 /dev/dsk/c4t0d0
```

SEE ALSO

pvcreate(1M), vgchange(1M), vgdisplay(1M), vgextend(1M), vgreduce(1M), lvm(7).

NAME

vgdisplay - display information about LVM volume groups

SYNOPSIS

/usr/sbin/vgdisplay [**-v**] [*vg_name* ...]

DESCRIPTION

The **vgdisplay** command displays information about volume groups. For each *vg_name* specified, **vgdisplay** displays information for that volume group only. If no *vg_name* is specified, **vgdisplay** displays names and corresponding information for all defined volume groups.

The volume group must be activated (see *vgchange*(1M)) before it can be displayed.

Options and Arguments

vgdisplay recognizes the following options and arguments:

vg_name
The path name of the volume group, for example, **/dev/vg00**.

-v For each volume group, display additional information about logical volumes, physical volumes, and physical volume groups.

Display Without –v Option

If you omit the **-v** option, only the following information is displayed:

--- Volume groups ---

VG Name	The path name of the volume group.
VG Write Access	Current access mode of the volume group. Possible values are **read/write** and **read-only**.
VG Status	State of the volume group: always **available**, as after a **vgchange -a y** command, since deactivated volume groups are not displayed.
Max LV	Maximum number of logical volumes allowed in the volume group.
Cur LV	Current number of logical volumes in the volume group.
Open LV	Number of logical volumes currently open in the volume group.
Max PV	Maximum number of physical volumes allowed in the volume group.
Cur PV	Current number of physical volumes in the volume group.
Act PV	Number of physical volumes that are currently active.
Max PE per PV	Maximum number (limit) of physical extents that can be allocated from any of the physical volumes in the volume group.
VGDA	Number of Volume Group Descriptor Areas within the volume group.
PE Size	Size of each physical extent.
Total PE	Total number of physical extents within the volume group: the sum of the number of physical extents belonging to each available physical volume in the volume group. (This does not include physical extents belonging to stand-by spare physical volumes; presence of these is only possible if you are using mirrored disks -- see below).
Alloc PE	Number of physical extents currently allocated to logical volumes.
Free PE	Number of physical extents not allocated (not including physical extents belonging to stand-by spares).
Total PVG	Total number of physical volume groups within the volume group.
Total Spare PVs	Total number of physical volumes that are designated as spares for this volume group. This will include both stand-by and active spares -- see below.

`Total Spare PVs in use`
Total number of spare physical volumes that are active in place of (containing all data from) a failed physical volume.

Display With −v Option

If you specify the **`-v`** option, **`vgdisplay`** lists the following additional information for each logical volume, for each physical volume, and for each physical volume group in the volume group:

`--- Logical volumes ---`

Information about logical volumes belonging to *vg_name*:

`LV Name` The block device path name of a logical volume in the volume group.

`LV Status` State of the logical volume:

`available/stale`	Logical volume available but contains physical extents that are not current.
`available/syncd`	Logical volume available and synchronized.
`available`	Logical volume available; **`stale`**/ **`syncd`** state cannot be confidently determined because logical volumes have both the Mirror Write Cache and Mirror Consistency Recovery turned off.
`unavailable`	Logical volume is not available for use.

`LV Size (Mbytes)`
Size of the logical volume.

`Current LE` Number of logical extents in the logical volume.

`Allocated PE` Number of physical extents used by the logical volume.

`Used PV` Number of physical volumes used by the logical volume.

`--- Physical volumes ---`

Information about physical volumes belonging to *vg_name*:

`PV Name` The block device path name of a physical volume in the group. When an alternate link to a physical volume has been added, **`Alternate Link`** is displayed next to the device path name. (See *vgextend*(1M) for definition.)

`PV Status` State of the physical volume: (*NOTE*: spare physical volumes are only relevant if you have installed HP MirrorDisk/UX software):

`available`	The physical volume is available and is not a spare physical volume.
`available/data spared`	The physical volume is available. However, it's data still resides on an active spare.
`available/active spare`	The physical volume is available and is an active spare physical volume. (An active spare is a spare that has taken over for a failed physical volume.)
`available/standby spare`	The physical volume is a spare "standing by" in case of a failure on any other physical volume in this volume group. It can only be used to capture data from a failed physical volume.
`unavailable`	The physical volume is unavailable and is not a spare physical volume.
`unavailable/data spared`	The physical volume is unavailable. However, it's data now resides on an active spare, and its data is available if the active spare is available.

unavailable/active spare
The physical volume is unavailable and it's an active spare. Thus, the data on this physical volume is unavailable.

unavailable/standby spare
The physical volume is a spare "standing by" that is not currently available to capture data from a failed physical volume.

Total PE Total number of physical extents on the physical volume.

Free PE Number of free physical extents on the physical volume.

Spared from PV
If the physical volume represents an active spare, this field will show the name of the failed physical volume whose data now resides on this spare. This information can be used to manually move the data back to the original physical volume once it has been repaired (see *pvmove*(1M)). If it cannot be determined which physical volume that the data came from, this field will instead display **Missing PV**. A missing PV would indicate that when the volume group was last activated or reactivated (see *vgchange*(1M)), the "failed" physical volume was not able to attach to the volume group.

Spared to PV If the physical volume represents a failed physical volume, this field will show the name of the active spare physical volume that now contains the data that originally residing on this volume. This information can be used to manually move the data back to the original physical volume (see *pvmove*(1M)) once it has been repaired.

Autoswitch For multiported devices accessed via multiple paths, this field indicates the *autoswitch* behavior for the physical volume (see *pvchange*(1M)).

On LVM will automatically switch from the path it is using whenever a better path to the physical volume is available. LVM will switch paths when a better path recovers (after it had failed earlier), or if the current path fails and another path is available. This is the default.

Off LVM will automatically switch to using the best available path only when the path currently in use is unavailable. LVM will continue using a specific path for the physical volume as long as it works, regardless of whether another better path recovers from a failure.

--- Physical volume groups ---

Information about physical volume groups belonging to *vg_name*:

PVG Name Name of a physical volume group in the volume group.

PV Name The block device path name of a physical volume in the physical volume group.

V

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages are displayed.

If **LANG** is not specified or is null, it defaults to "C" (see *lang*(5)).

If any internationalization variable contains an invalid setting, all internationalization variables default to "C" (see *environ*(5)).

EXAMPLES

Display information about all the volume groups within the system:

```
vgdisplay
```

Display all of the information about one volume group, including the characteristics and status of both the logical and physical extents of the volume group:

```
        vgdisplay -v /dev/vg02
```

SEE ALSO

lvdisplay(1M), pvdisplay(1M), vgchange(1M), vgcreate(1M).

NAME

vgexport - export an LVM volume group and its associated logical volumes

SYNOPSIS

/usr/sbin/vgexport [**-m** *mapfile*] [**-p**] [**-v**] [**-f** *outfile*] *vg_name*

/usr/sbin/vgexport -m *mapfile* **-s -p -v** *vg_name*

DESCRIPTION

Using the format of the first command line of the *SYNOPSIS* above, the **vgexport** command can be used to remove a volume group from the system. The volume group will be removed without modifying the logical volume information found on the physical volumes.

The volume group identified by *vg_name* is removed from the **/etc/lvmtab** file, and the associated device files including the *vg_name* directory and **group** file are removed from the system.

The volume group information and data is untouched on the physical volume. These disks can be imported to other system with the **vgimport** command (see *vgimport*(1M)).

Sharable Option, Series 800 Only

Using the format of the second command line of the *SYNOPSIS* above, the **vgexport** command generates a *mapfile* that can be copied to other systems that are part of a high availability cluster and the **vgimport** command (see *vgimport*(1M)) can be used to recreate the volume group. See also *vgchange*(1M). The *mapfile* contains a description of the volume group and its associated logical volume(s) (if any). The logical volume information found on the physical volumes is not modified. Note that with this option, the volume group is not removed from the system. (See the second example below).

The volume group specified in the *mapfile* can be shared with the importing systems. The volume group is not removed from the exporting system.

Options and Arguments

vgexport recognizes the following options and arguments:

vg_name	The path name of the volume group.
-m *mapfile*	By default, a file named **mapfile** is created in the current directory. This file contains a description of the volume group and its associated logical volume(s) (if any). Use this option to specify a different name for the file, *mapfile*. This file can be used as input to **vgimport** (see *vgimport*(1M)). When used with the **-s** option, the volume group specified in the *mapfile* can be shared with other systems in the high availability cluster.
-p	Preview the actions to be taken but do not update the **/etc/lvmtab** file or remove the devices file. This option is best used in conjunction with the **-v** option.
-v	Print verbose messages including the names of the physical volumes associated with this volume group.
-s	Sharable option, Series 800 only. When the **-s** option is specified, then the **-p**, **-v**, and **-m** options must also be specified. A *mapfile* is created that can be used to create volume group entries on other systems in the high availability cluster (with the **vgimport** command).
-f *outfile*	Write the current set of *pv_paths* for the volume group to the *outfile*. The *outfile* may then be used as the *infile* for the **vgimport -f** option. If used together with the **-p** option the volume group is not exported but the list of *pv_paths* is still written to the *outfile*. This may be useful to derive a list of *pv_paths* for the volume group or to use on another system which is sharing the volume group and which has an identical configuration.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages are displayed.

If **LANG** is not specified or is null, it defaults to "C" (see *lang*(5)).

If any internationalization variable contains an invalid setting, all internationalization variables default to "C" (see *environ*(5)).

EXAMPLES

Export the volume group **/dev/vg01** into *mapfile* **vg01.mymap**. The volume group will be removed from the exporting system.

```
vgexport -m  vg01.mymap /dev/vg01
```

Export the volume group **/dev/vg01** and write the disk names into the file **vg01.outfile**.

```
vgexport -v -f outfile /dev/vg01
```

Create a *mapfile* to be copied to other systems in a high availability cluster to build the volume group information for the volume group, **/dev/vg02**. Note that the volume group is not removed from the exporting system. The importing systems will create the volume group with the **vgimport** command using the **-s** and **-m** options.

```
vgexport -s -p -m -v  vg02.mymap /dev/vg02
```

SEE ALSO

vgimport(1M), vgscan(1M).

NAME

vgextend - extend an LVM volume group by adding physical volumes

SYNOPSIS

/usr/sbin/vgextend [**-f**] [**-A** *autobackup*] [**-g** *pvg_name*] [**-x** *extensibility*] [**-z** *sparepv*]
vg_name pv_path ...

Remarks

vgextend cannot be performed if the volume group is activated in shared mode.

DESCRIPTION

The **vgextend** command assigns additional physical volumes to volume group *vg_name*. The volume group must be active.

Volume groups are extended by adding one or more physical volumes specified by *pv_path* ...

After the physical volumes have been successfully added to the volume group, the disk space they contain can be allocated to logical volumes.

Before assigning an additional physical volume to a volume group, create the physical volume with the **pvcreate** command (see *pvcreate*(1M)). Then, create the volume group with the **vgcreate** command, assigning at least one physical volume (see *vgcreate*(1M)).

If, for any reason, a specified physical volume cannot be installed into the volume group, an error message is displayed. However, the installation continues to the end of the list of physical volumes.

When a *pv_path* refers to one of the physical volumes already in the volume group by a different *pv_path* name to indicate the use of a different controller, this new path becomes an **alternate link** to the physical volume. When two paths that reference the same disk are provided in the *pv_path* list, the order of the paths is important. The first path becomes the "primary link" to the physical volume, the second becomes an "alternate link" to the physical volume. The primary link is the path used to access the physical volume unless the primary link becomes unavailable in which case LVM automatically switches to the alternate link to access the physical volume. Currently LVM supports a maximum of 8 paths to a physical volume (7 alternates and one primary).

Options and Arguments

vgextend recognizes the following options and arguments:

pv_path The block device path name of a physical volume.

vg_name The path name of the volume group.

-A *autobackup* Set automatic backup for this invocation of this command. *autobackup* can have one of the following values:

- **y** Automatically back up configuration changes made to the volume group. This is the default.

  After this command executes, the **vgcfgbackup** command (see *vgcfgbackup*(1M)) is executed for the volume group.

- **n** Do not back up configuration changes this time.

-f This option will force a volume group to be extended with a physical volume which has alternate block(s) already allocated, (i.e. this physical volume was not initialized using **pvcreate -f**.) This option should be used with extreme caution. If the disk is being extended to a volume group with a different physical extent size, the alternate block(s) might be inside the user data area. Potential data corruption could occur.

-g *pvg_name* Extend an existing physical volume group while the volume group is being extended by adding all the physical volumes in the *pv_path* parameter to the physical volume group specified by *pvg_name*.

If the specified physical volume group does not exist, it is created, thus providing a means for creating new physical volume groups after the volume group has been created. Another way to extend or add a physical volume group is to edit the **/etc/lvmpvg** file as described in *vgcreate*(1M). See *lvmpvg*(4) for format details.

-x *extensibility* Set allocation permission for additional physical extents on the physical volume specified by *pv_path*. *extensibility* can have one of the following values:

y Allow allocation of additional physical extents on the physical volume.

n Prohibit allocation of additional physical extents on the physical volume. Logical volumes residing on the physical volume can still be accessed.

-z *sparepv* This option requires the installation of the optional HP MirrorDisk/UX software. It allows you to mark the physical volume(s) specified by *pv_path* to be either a spare physical volume or a regular, non-spare physical volume. (A spare physical volume can be used to replace an existing physical volume within a volume group when mirroring is in effect, in the event the existing physical volume fails.) *sparepv* can have one of the following values:

y The physical volume(s) will be used as spare(s). No physical extents from a spare physical volume will be available as part of the "free" pool of extents in the volume group. The spare physical volume(s) will only be used in the event of another physical volume within this volume group becomes unavailable (fails).

n The physical volume(s) will be used as regular, non-spare members of the volume group. This is the default.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages are displayed.

If **LANG** is not specified or is null, it defaults to "C" (see *lang*(5)).

If any internationalization variable contains an invalid setting, all internationalization variables default to "C" (see *environ*(5)).

EXAMPLES

Add physical volumes **/dev/dsk/c0t1d0** and **/dev/dsk/c0t2d0** to volume group **/dev/vg03**:

```
vgextend /dev/vg03 /dev/dsk/c0t1d0 /dev/dsk/c0t2d0
```

Extend physical volume group **PVG0** while adding physical volumes **/dev/dsk/c0t3d0** and **/dev/dsk/c0t4d0** to volume group **/dev/vg03**:

```
vgextend -g PVG0 /dev/vg03 /dev/dsk/c0t3d0 /dev/dsk/c0t4d0
```

Add an alternate link to one of the physical volumes in the volume group where **/dev/dsk/c0t4d0** and **/dev/dsk/c1t4d0** refer to the same physical volume (referenced via different controllers), and the volume group already contains **/dev/dsk/c0t4d0**. **/dev/dsk/c0t4d0** remains the primary link (in use) and **/dev/dsk/c1t4d0** becomes the alternate link.

```
vgextend  /dev/vg03 /dev/dsk/c1t4d0
```

Add a spare physical volume to a volume group:

```
vgextend  -z y /dev/vg03 /dev/dsk/c2t4d0
```

WARNINGS

The new physical volume which has been added to the volume group could potentially have a different block size compared to physical volumes already in the volume group.

If a logical volume is created on two or more physical volumes which have a different block size, it is not possible to use such logical volume for file system purposes. See *extendfs*(1M).

For example, when a logical volume contains physical volumes that all have 1k block size, and then it is extended to contain a physical volume with 2k block size, then the block size of the volume group is increased to 2k.

ADDITIONAL WARNINGS

Starting at 10.20, the **vgextend** command is supporting additional EMC Symmetrix disk product functionality (see *vgchgid*(1M)). The **vgextend** command will be enforcing a new rule such that a physical device, to be extended to a volume group, must have the same EMC Symmetrix attributes as the physical volumes already in the volume group. Clearly this checking only applies if the disks involved (those already in the volume group and those being added) in the operation are EMC Symmetrix. Should the command find an incompatibility, a message of the following type will be produced:

> "Attempt to mix incompatible EMC Symmetrix disk types failed. Volume group contains standard disks, but physical volume /dev/dsk/cxtxdx contains EMC BCV DISK. Please consult the EMC Symmetrix documentation, if applicable".

Cross-Cabinet Volume Group Warnings

On multi-cabinet V-Class systems, it is possible to create a volume group using physical volumes that are physically attached to different cabinets. However, due to limitations in the HP-UX boot sequence, such a cross-cabinet volume group cannot contain root, boot, swap, or dump logical volumes. If the volume group *vg_name* contains a root, boot, swap, or dump logical volume, and by adding the physical volume(s) specified by *pv_path* to the volume group it would become cross-cabinet, then **vgextend** will fail.

SEE ALSO

pvchange(1M), pvcreate(1M), vgchange(1M), vgcreate(1M), vgdisplay(1M).

NAME

vgimport - import an LVM volume group onto the system

SYNOPSIS

/usr/sbin/vgimport [**-m** *mapfile*] [**-p**] [**-v**] [**-f** *infile*] *vg_name pv_path* ...

/usr/sbin/vgimport -m *mapfile* **-s -v** *vg_name*

DESCRIPTION

The **vgimport** command adds the specified volume group to the system. The physical volumes, specified as *pv_path* ..., are scanned to obtain the volume group information and logical volume information. This command works much like **vgcreate** by requiring that the volume group device directory and the **group** special file be created before the command is executed (see *vgcreate*(1M)). The *vg_name* is added from the **/etc/lvmtab** file, and the associated logical volume device files are added to the system.

vgimport assumes that the volume group information has already been created on the physical volumes.

This command is useful in conjunction with the **vgexport** command (see *vgexport*(1M)), to move volume groups from one system to other systems within a high availability cluster.

vgimport creates logical volume devices files under the *vg_name* directory using the naming convention given in *mapfile* or using the default naming convention used by the **lvcreate** command (see *lvcreate*(1M)).

Sharable Option, Series 800 Only

In the second format of the command line shown in **SYNOPSIS**, the volume group specified in the *mapfile* is shared.

vgimport does not activate the imported volume group due to the many possible options at volume group activation time. To activate the volume group once it has been successfully imported, use the **vgchange** command (see *vgchange*(1M)).

Options and Arguments

vgimport recognizes the following options and arguments:

pv_path	The block device path names of a physical volume. This argument is not used with the **-s** and related options.
vg_name	The path name of the volume group.
-m *mapfile*	Specify the name of the file from which logical volume names and numbers are to be read. This option is optional when used as in the first command line format of the *SYNOPSIS*. If this option is not specified, logical volume names are created using the default naming convention **lvol***nn* where *nn* is the logical volume minor number. When used with the **-s** option, the volume group specified in the *mapfile* can be shared among the exporting system and the importing systems.
-s	Sharable option, Series 800 only. When the **-s** option is specified, then the **-p**, **-v**, and **-m** options must also be specified. The specified *mapfile* is the same *mapfile* specified by using the **vgexport** command also using the **-p**, **-m**, and **-s** options. The *mapfile* is used to create the volume groups on the importing systems.
-p	Preview the actions to be taken but do not update the **/etc/lvmtab** file or add the devices file. This option is best used in conjunction with the **-v** option.
-v	Print verbose messages including names of the logical volumes.
-f *infile*	Import the set of *pv_paths* held in the *infile* into the volume group. This option is used as an alternative to specifying the *pv_paths* on the command line. Each *pv_path* must appear on a new line in the *infile*. This option may not be used together with the **-s** option.

WARNINGS

The following warnings should only apply to the **-s** option when importing devices such as (NIKE) or disks with alternate path:

Since the **-s** option causes a search on the system for each disks with the same **vg_id.** When **vgimport** reconstruct the newly imported volume group entry in **/etc/lvmtab** file, the order of disks could be different than it was before. And the following will happen:

The designated primary and alternate link might not be the same as it was configured before.

Alternate links will be added to the importing volume group even if they might not be configured in the volume group initially.

If the original primary path of a disks become an alternate path after the newly imported volume group entry is created in **/etc/lvmtab,** the order can be easily reverted by using **vgreduce** to remove the primary path and then use **vgextend** to add the path back again.

If additional alternate paths were added to the newly imported volume group, use **vgreduce** to reduce any alternate paths that were added but they were not needed.

ADDITIONAL WARNINGS

Starting at 10.20, the **vgimport** command is supporting additional EMC Symmetrix disk product functionality (see *vgchgid*(1M)). The **vgimport** command will be enforcing a new rule such that the group of disks to be imported must have the same EMC Symmetrix attributes. Clearly this checking only applies if the disks involved in the operation are EMC Symmetrix. Should the command find an incompatibility, a message of the following type will be produced:

"Attempt to mix incompatible EMC Symmetrix disk types failed. Physical volume /dev/dsk/cxtxdx is EMC BCV disk, but physical volume /dev/dsk/cxtxdx is a standard disk. Please consult the EMC Symmetrix documentation, if applicable".

If duplicate VGIDs (volume group id) were found in different types of EMC Symmetrix disks during the **vgimport -s** operation, the above error message will appear, and the command will fail. Please refer to *vgchgid*(1M) on how to modify VGID on the EMC Symmetrix disk.

Note that EMC Symmetrix disks which do not have any of the additional attributes can still be mixed with non-EMC disks in a single volume group.

Cross-Cabinet Volume Group Warnings

On multi-cabinet V-Class systems, it is possible to create a volume group using physical volumes that are physically attached to different cabinets. However, due to limitations in the HP-UX boot sequence, such a cross-cabinet volume group cannot contain root, boot, swap, or dump logical volumes. If the physical volumes specified by *pv_path* make up a cross-cabinet volume group that contains a root, boot, swap, or dump logical volume, then **vgimport** will fail to import that volume group. This should only occur as the result of a hardware reconfiguration. If this does occur, the hardware configuration will need to be modified again to connect the physical volumes to the same cabinet.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages are displayed.

If **LANG** is not specified or is null, it defaults to "C" (see *lang*(5)).

If any internationalization variable contains an invalid setting, all internationalization variables default to "C" (see *environ*(5)).

EXAMPLES

Import the volume group **/dev/vg01** that is located on physical disks **/dev/dsk/c0t1d0** and **/dev/dsk/c0t3d0**:

```
vgimport -v /dev/vg01 /dev/dsk/c0t1d0 /dev/dsk/c0t3d0
```

Activate the volume group following a successful import:

```
vgchange -a y vg01
```

Import the volume group **/dev/vg01** using the *mapfile*, **/tmp/vg01.mymap**. **mymap** was previously specified by the **vgexport** command on another system. The volume group, **/dev/vg01**, is specified in **mymap** and will be used by the importing system only:

```
vgimport -v -m /tmp/vg01.mapfile /dev/vg01 \
/dev/dsk/c0t5d0 /dev/dsk/c0t7d0
```

Import the volume group **/dev/vg02** using the *mapfile*, **/tmp/vg02.mymap**. **mymap** was previously specified by the **vgexport** command on another system. The volume group, **/dev/vg02**, is specified in **mymap** and will be shared among the exporting system, this system, and other systems importing the volume group as shared:

V

```
    vgimport -v -s -m /tmp/vg02.mymap /dev/vg02
```

Import the volume group **/dev/vg02** using the *infile*, **/tmp/vg02.infile**. **infile** was previously specified by the **vgexport** command on another system.

```
    vgimport -v -f /tmp/vg02.infile /dev/vg02
```

SEE ALSO

vgexport(1M), vgscan(1M).

NAME

vgreduce - remove physical volumes from an LVM volume group

SYNOPSIS

/usr/sbin/vgreduce [**-A** *autobackup*] *vg_name pv_path* ...

/usr/sbin/vgreduce [**-A** *autobackup*] [**-l**] *vg_name pv_path*

/usr/sbin/vgreduce [**-A** *autobackup*] [**-f**] *vg_name*

Remarks

vgreduce cannot be performed if the volume group is activated in shared mode.

DESCRIPTION

The **vgreduce** command removes each physical volume specified by a *pv_path* argument from volume group *vg_name*.

The **vgreduce** command with -f option removes all missing physical volume from the volume group.

All but one physical volume can be removed. The last physical volume must remain in the volume group so that the logical volume driver can continue to operate. The last physical volume in the volume group can be removed with the **vgremove** command (see *vgremove*(1M)).

Before executing **vgreduce**, remove all logical volumes residing on each physical volume represented by a *pv_path* by executing **lvremove** (see *lvremove*(1M)).

Any physical volume in the *pv_path* list that is also a member of a physical volume group (as defined in **/etc/lvmpvg**) is also removed from the physical volume group. If the physical volume happens to be the last one in the physical volume group, the physical volume group is also removed from the volume group.

When a physical volume in the *pv_path* list has multiple **PV-links**, the physical volume is *not* removed from the volume group, until all the links to the volume are removed. When a physical volume in the *pv_path* list is the **primary link** (in use) to a physical volume, removing the link forces LVM to switch to the **alternate link** to access the physical volume. When the *pv_path* removed is an **alternate link** to the device, only the link is removed; the volume group and physical volume are otherwise unchanged.

Options and Arguments

vgreduce recognizes the following options and arguments:

-A *autobackup*

Set automatic backup for this invocation of this command. *autobackup* can have one of the following values:

y Automatically back up configuration changes made to the volume group. This is the default.

After this command executes, the **vgcfgbackup** command (see *vgcfgbackup*(1M)) is executed for the volume group.

n Do not back up configuration changes this time.

-f *vg_name* force reduction of missing physical volume(s) in a given volume group. This option does not require a physical volume (PV) to be supplied on the command line. **vgreduce** obtains the name of each physical volume (PV) belonging to the volume group from the file **/etc/lvmtab**. It then reads the LVM structures from each PV and compares these with that held by the kernel to work out which PVs are missing. PVs which are missing will be candidates for removal. If all the physical extents on the missing PV are free then it will be removed from the volume group. Otherwise **vgreduce** will report the physical to logical extent mapping. For missing PVs, which have extents in use, you must free up all the extents by using *lvreduce*(1M) or *lvremove*(1M) and re-run **vgreduce** with the **-f** option. This option is most commonly used when the *vgdisplay*(1M) command shows "Cur PV" higher than "Act PV" and all of the PVs belonging to the volume group are attached. This option only works on PVs and not on links.

-l *pv_path* This option removes the specified *pv_path/s* from the **lvmtab** file. This task will only be performed if the *pv_path* is currently marked as missing from the volume group. This option was mainly designed for the problem where the **vgscan** or the **vgimport** command place too many links beyond the max limit allowed in the

lvmtab file. Currently the max limit is 8 paths to a PV (7 alternates and one primary). In this situation invoking the command without the **-f** option will not resolve the condition because the path is not attached to the volume group. Similarly the condition would not be overcome by invoking with the **-f** option as this works on PVs rather than links.

pv_path The block device path name of a physical volume.

vg_name The path name of the volume group.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages are displayed.

If **LANG** is not specified or is null, it defaults to "C" (see *lang*(5)).

If any internationalization variable contains an invalid setting, all internationalization variables default to "C" (see *environ*(5)).

EXAMPLES

Remove physical volume **/dev/dsk/c0t1d0** from volume group **/dev/vg01**:

```
vgreduce /dev/vg01 /dev/dsk/c0t1d0
```

Force reduction of missing PVs from volume group: **vg01**

```
vgreduce -f /dev/vg01
```

Removes the physical volume from the volume group when it is still in the **lvmtab** file, but it is currently marked as missing from the volume group: **vg01**

```
vgreduce -l /dev/vg01
```

The following messages will appear after missing PVS has been removed successfully:

PV with key 0 successfully deleted from vg **/dev/vg01**

Repair done, please do the following steps.....:

1. Save **/etc/lvmtab** to another file.
2. Remove **/etc/lvmtab**.
3. Use **vgscan -v** to recreate **/etc/lvmtab**.
4. NOW use *vgcfgbackup*(1M) to save the LVM setup.

SEE ALSO

vgchange(1M), vgcreate(1M), vgdisplay(1M), vgextend(1M).

NAME

vgremove - remove LVM volume group definition from the system

SYNOPSIS

/usr/sbin/vgremove *vg_name* ...

DESCRIPTION

The **vgremove** command removes from the system the last physical volume of the volume group and the definition of the volume group or groups specified by *vg_name* .... Since all system knowledge of the volume group and its contents are removed, the volume group can no longer be accessed.

To move a volume group from one system to another, use the **vgexport** command instead (see *vgexport*(1M)).

Before executing **vgremove**, remove all logical volumes residing on the last physical volume by executing **lvremove** (see *lvremove*(1M)).

vgremove is equivalent to the inverse of executing **vgcreate** for one physical volume (see *vgcreate*(1M)).

Before removing a volume group, two steps are necessary:

1. Remove all the logical volumes belonging to the group by using the **lvremove** command (see *lvremove*(1M)).
2. Remove all but one physical volume belonging to the volume group by using the **vgreduce** command (see *vgreduce*(1M)).

If there is any physical volume group created under *vg_name* ..., the physical volume group information is also removed from file **/etc/lvmpvg**.

Arguments

vgremove recognizes the following argument:

vg_name	The path name of a volume group.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages are displayed.

If **LANG** is not specified or is null, it defaults to "C" (see *lang*(5)).

If any internationalization variable contains an invalid setting, all internationalization variables default to "C" (see *environ*(5)).

EXAMPLES

Remove volume group **/dev/vg02** from the system:

```
vgremove /dev/vg02
```

SEE ALSO

lvremove(1M), vgchange(1M), vgreduce(1M).

NAME

vgscan - scan physical volumes for LVM volume groups

SYNOPSIS

/usr/sbin/vgscan [**-a**] [**-p**] [**-v**]

DESCRIPTION

The **vgscan** command allows the re-creation of the **/etc/lvmtab** file and possibly the associated volume group device files. This command should be run only in the event of a catastrophic error such as the deletion of the **/etc/lvmtab** file or the mismatch of names of the physical volumes in the **/etc/lvmtab** file to the actual physical volume path configuration. If the **/etc/lvmtab** file exists, the information contained in the file is used to assist in rebuilding the file, but the existing file is updated with the new corrected configuration.

vgscan searches each physical volume connected to the system, looking for logical volumes. If there are dual controller devices, only the primary controller device path is scanned, unless you specify the **-a** option to allow access to all paths. It groups these physical volumes into volume groups by matching the volume group information found on the physical volumes. Then it searches the **/dev** directory for all **group** device files with the LVM major number, and tries to match device files with the logical volumes' information found on the physical volumes.

If matches occur, it determines the volume group name from the device file path, and updates the **/etc/lvmtab** file with the volume group name and the list of physical volumes paths contained in that volume group. For volume groups where the device files cannot be matched, it prints the list of physical volumes for each volume group.

After **vgscan** completes successfully, run the **vgimport** command on each set of unmatched physical volumes (see *vgimport*(1M)).

Options

vgscan recognizes the following options:

-a Scan all controller device paths for all disks.

-p Preview the actions that would be taken but do not update file **/etc/lvmtab**. This option is best used in conjunction with the **-v** option.

-v Print verbose messages.

WARNINGS

The following warning only applies to dual controller devices (NIKE), or disks with alternate path:

Since **vgscan** search each disks on the system in the order of where they have configured. When **vgscan** reconstructs the **/etc/lvmtab** file, the order of disks in the file could be different than it was before. The following will happen:

The designated primary and alternate link might not be the same as it was configured before.

Alternate links will be added to the **/etc/lvmtab** file even if they might not be configured in the volume group initially.

The boot information might be incorrect due to different order of disks in the new **/etc/lvmtab** file.

In order to correct the above problems, do the following:

Use **vgchange** with **-a** option to activate all volume groups.

Use **lvlnboot** with -R option to correct boot information on disk.

Use **vgreduce** to reduce any alternate links that were added to the **/etc/lvmtab** file by **vgscan**, but they were not needed.

If the original primary path of a disks become an alternate path after **/etc/lvmtab** file is reconstructed, the order can be easily reverted by using **vgreduce** to remove the primary path and use **vgextend** to add the path back again.

If **/etc/lvmtab** is destroyed, do not use **vgscan** to re-construct **/etc/lvmtab** if the system is heavily loaded by an application. Otherwise, **vgscan** will create an incomplete **/etc/lvmtab** due to a known NIKE/LVM limitation issue. It's important to quiesce the logical volume's I/O before re-constructing the **/etc/lvmtab**.

If for some reason, there is a need to re-construct **/etc/lvmtab** when the system is running production application, **vgscan** will create a partial **/etc/lvmtab**. In this case, most of the primary paths should be included in the **/etc/lvmtab**. Use **vgextend** to include any missing alternate paths in the VG.

EMC DISK WARNINGS

Starting at 10.20, the **vgscan** command is supporting additional EMC Symmetrix disk product functionality (see *vgchgid*(1M)). The **vgscan** command will be enforcing a new rule such that only EMC disks with the same attributes can belong to the same volume group. Note that an EMC disk, without any of the additional attributes, can still be mixed with non-EMC disks in the same volume group. Please consult your EMC Symmetrix documentation, if applicable.

CROSS-CABINET VOLUME GROUP WARNINGS

On multi-cabinet V-Class systems, it is possible to create a volume group using physical volumes that are physically attached to different cabinets. However, due to limitations in the HP-UX boot sequence, such a cross-cabinet volume group cannot contain root, boot, swap, or dump logical volumes. If **vgscan** discovers a cross-cabinet volume group that contains a root, boot, swap, or dump logical volume, it will not update the **/etc/lvmtab** file for that volume group.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages are displayed.

If **LANG** is not specified or is null, it defaults to "C" (see *lang*(5)).

If any internationalization variable contains an invalid setting, all internationalization variables default to "C" (see *environ*(5)).

EXAMPLES

Scan the primary controller device paths for all the physical volumes on the system, but do not update the **/etc/lvmtab** file:

```
vgscan -p -v
```

Scan all controller device paths to all physical volumes on the system:

```
vgscan -a -v
```

Scan the primary controller device paths for all the physical volumes on the system and re-create **/etc/lvmtab** file:

```
vgscan -v
```

The following messages will appear after **/etc/lvmtab** file is re-created.

```
*** LVMTAB has been created successfully.
*** If PV links are configured in the system
*** Do the following to resync information on disk.
*** #1.  vgchange -a y
*** #2.  lvlnboot -R
```

SEE ALSO

vgexport(1M), vgimport(1M).

(Requires Optional HP MirrorDisk/UX Software)

NAME

vgsync - synchronize stale logical volume mirrors in LVM volume groups

SYNOPSIS

/usr/sbin/vgsync *vg_name* ...

Remarks

This command requires the installation of the optional HP MirrorDisk/UX software, which is not included in the standard HP-UX operating system.

DESCRIPTION

The **vgsync** command synchronizes the physical extents of each mirrored logical volume in the volume group specified by *vg_name* .... Synchronization occurs only on the physical extents that are stale mirrors of the original logical extent.

The synchronization process can be time consuming, depending on the hardware characteristics and the amount of data. Unless disabled, the mirrors within a volume group are synchronized automatically when the volume group is activated by the **vgchange -a y** command.

Arguments

vgsync recognizes the following argument:

vg_name	The path name of a volume group.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages are displayed.

If **LANG** is not specified or is null, it defaults to "C" (see *lang*(5)).

If any internationalization variable contains an invalid setting, all internationalization variables default to "C" (see *environ*(5)).

EXAMPLES

Synchronize the mirrors on volume group **/dev/vg04**:

```
vgsync /dev/vg04
```

WARNINGS

It is not advisable to interrupt a **vgsync** process.

SEE ALSO

lvsync(1M), vgchange(1M), vgdisplay(1M).

NAME

vipw - edit the password file

SYNOPSIS

vipw

DESCRIPTION

vipw edits the password file while setting the appropriate locks, and does any necessary processing after the password file is unlocked. If the password file is already being edited, you will be told to try again later. The **vi** editor is used unless the environment variable **EDITOR** indicates an alternate editor.

vipw performs a number of consistency checks on the password entry for **root**, and does not allow a password file with an incorrectly formatted root entry to be installed. To help reduce the possibility of leaving the system in an unbootable state, root's entry is not considered properly formatted if it has a user id that is not zero, or if it has a shell other than **/sbin/sh**, **/usr/bin/csh**, **/usr/bin/ksh**, or **/usr/bin/sh**.

Please refer to *passwd*(4) and the *Managing Systems and Workgroups* manual for further details of password file format.

WARNINGS

An **/etc/ptmp** file not removed when a system crashes prevents further editing of the **/etc/passwd** file using **vipw** after the system is rebooted. **/etc/ptmp** is the standard lock used by all commands which knowingly modify **/etc/passwd**.

Successful execution of **vipw** is not sufficient for proper system operation. To help maintain consistency with other system databases, editing of the password file with **vipw** is generally discouraged. Please use **sam**, **useradd**, **usermod**, **userdel**, **chfn**, **chsh** or **passwd** to edit **/etc/passwd.**

AUTHOR

vipw was developed by the University of California, Berkeley.

FILES

/etc/ptmp

SEE ALSO

passwd(1), sam(1M), useradd(1M), usermod(1M), userdel(1M), chfn(1), chsh(1), passwd(4).

NAME

volcopy, labelit - copy a file system with label checking

SYNOPSIS

/usr/sbin/volcopy [*options*] *fsname special1 volname1 special2 volname2*

/usr/sbin/labelit [*options*] *special* [*fsname volume* [**-n**]]

DESCRIPTION

The **volcopy** command makes a literal copy of the file system using a block size matched to the device.

Options

volcopy recognizes the following options:

-F *FStype* Specify the file system type on which to operate (see *fstyp*(1M) and *fs_wrapper*(5)). If this option is not included on the command line, then the file system type is determined from the file **/etc/fstab** by matching *special* with an entry in that file. If there is no entry in **/etc/fstab**, then the file system type is determined from the file **/etc/default/fs**.

-a Invoke a verification sequence requiring a positive operator response instead of the standard delay before the copy is made.

-o *specific_options*
Specify options specific to the file system type. *specific_options* is a list of suboptions and/or keyword/attribute pairs intended for an *FStype*-specific module of the command. See the file system specific manual entries for a description of the *specific_options* that are supported, if any.

-s (default) Invoke the DEL-if-wrong verification sequence.

-y Assume a **yes** response to all questions

-V Echo the completed command line, but perform no other actions. The command line is generated by incorporating the user-specified options and arguments with other information derived from **/etc/fstab**. This option allows the user to verify the command line.

Other options are used with 9-track magnetic tapes:

-bpi *density*	Bits per inch.
-feet *size*	Size of reel in feet.
-reel *num*	Beginning reel number for a restarted copy.
-buf	Use double buffered I/O.

The **volcopy** command requests length and density information if they are not given on the command line and they are not recorded on an input tape label. If the file system is too large to fit on one reel, the **volcopy** command prompts for additional reels. Labels of all reels are checked. Tapes can be mounted alternately on two or more drives. If the **volcopy** command is interrupted, it asks if the user wants to quit or wants to escape to the command interpreter. In the latter case, other operations (such as executing the **labelit** command) can be performed before returning to the **volcopy** command by exiting the command interpreter.

fsname	The file system name on the device (e.g., **root**) being copied.
special	The physical disk section or tape (e.g., **/dev/rdsk/1s3** or **/dev/rmt/c0t0d0BEST**).
volname	The physical volume name; it should match the external sticker. Such label names are limited to six or fewer characters. The argument *volname* can be **-** to use the existing volume name.
special1	The device from which the copy of the file system is being extracted.
volname1	The volume from which the copy of the file system is being extracted.
special2	The target device.
volname2	The target volume.

The **labelit** command can be used to provide initial labels for unmounted disk or tape file systems. With the optional arguments omitted, the **labelit** command prints current label values. The **-n** option provides for initial labeling of new tapes only (this destroys previous contents). The **-F**, **-V**, and **-o** options can be specified for the **labelit** command. The behavior of the **-F**, **-V**, and **-o** options is similar to their behavior in the **volcopy** command.

FILES

/etc/default/fs	File that specifies the default file system type.
/etc/fstab	Static information about the file systems.

SEE ALSO

volcopy_hfs(1M), volcopy_vxfs(1M), fs_wrapper(5).

NAME

volcopy, labelit - copy an HFS file system with label checking

SYNOPSIS

/usr/sbin/volcopy [*options*] *fsname special1 volname1 special2 volname2*

/usr/sbin/labelit [*options*] *special* [*fsname volume* [**-n**]]

DESCRIPTION

The **volcopy** command makes a literal copy of an HFS file system using a block size matched to the device.

Options

volcopy recognizes the following options:

-F hfs	Specifies the HFS file system type.
-a	Invoke a verification sequence requiring a positive operator response instead of the standard delay before the copy is made.
-s	(default) Invoke the DEL-if-wrong verification sequence.
-y	Assume a **yes** response to all questions
-V	Echo the completed command line, but perform no other actions. The command line is generated by incorporating the user-specified options and arguments with other information derived from **/etc/fstab**. This option allows the user to verify the command line.

Other options are used with 9-track magnetic tapes:

-bpi *density*	Bits per inch.
-feet *size*	Size of reel in feet.
-reel *num*	Beginning reel number for a restarted copy.
-buf	Use double buffered I/O.

The **volcopy** command requests length and density information if they are not given on the command line and they are not recorded on an input tape label. If the file system is too large to fit on one reel, the **volcopy** command prompts for additional reels. Labels of all reels are checked. Tapes can be mounted alternately on two or more drives. If the **volcopy** command is interrupted, it asks if the user wants to quit or wants to escape to the command interpreter. In the latter case, other operations (such as executing the **labelit** command) can be performed before returning to the **volcopy** command by exiting the command interpreter.

fsname	The file system name on the device (e.g., **root**) being copied.
special	The physical disk section or tape (e.g., **/dev/rdsk/1s3** or **/dev/rmt/c0t0d0BEST**).
volname	The physical volume name; it should match the external sticker. Such label names are limited to six or fewer characters. The argument *volname* can be **-** to use the existing volume name.
special1	The device from which the copy of the file system is being extracted.
volname1	The volume from which the copy of the file system is being extracted.
special2	The target device.
volname2	The target volume.

The **labelit** command can be used to provide initial labels for unmounted disk or tape file systems. With the optional arguments omitted, the **labelit** command prints current label values. The **-n** option provides for initial labeling of new tapes only (this destroys previous contents). The **-F** and **-V** options can be specified for the **labelit** command. The behavior of the **-F** and **-V** options is similar to their behavior in the **volcopy** command.

SEE ALSO

fstyp(1M), volcopy(1M), fs_wrapper(5).

NAME

volcopy, labelit - copy a VxFS file system with label checking

SYNOPSIS

/usr/sbin/volcopy [**-F vxfs**] [**-V**] [**-a**] [**-s**] [**-y**] *fsname special1 volname1 special2 volname2*

/usr/sbin/labelit [**-F vxfs**] [**-V**] [**-n**] *special* [*fsname volume*]

DESCRIPTION

volcopy makes a literal copy of a VxFS file system using a block size matched to the device.

Options

volcopy recognizes the following options and command-line arguments:

-F vxfs	Specify the file-system type (**vxfs**).
-V	Validate the command line options, but do not execute the command. If the options specified are valid, **-V** echoes the complete command line. If the options specified are not valid, **-V** prints an error message.
-a	Normally, before copying the file system, **volcopy** delays so that the user can interrupt the copying if needed. This option invokes a verification sequence requiring a positive operator response instead of the standard delay.
-s	(default) Before copying the file system, prints a message and allows the user to interrupt the copy if needed.
-y	Assume a **yes** response to all questions

Other options are used with 9-track magnetic tapes:

-bpi *density*	Bits per inch.
-feet *size*	Size of reel in feet.
-reel *num*	Beginning reel number for a restarted copy.
-buf	Use double buffered I/O.

The **volcopy** command requests length and density information it is not given on the command line or if it is not recorded on an input tape label. If the file system is too large to fit on one reel, **volcopy** prompts for additional reels. Labels of all reels are checked. Tapes can be mounted alternately on two or more drives. If **volcopy** is interrupted, it asks if the user wants to quit or to escape to the command interpreter. In the later case, other operations (such as **labelit**) can be performed before returning to **volcopy** by exiting the command interpreter.

The *fsname* argument represents the file system name on the device (e.g., **root**) being copied.

special should be the physical disk section or tape (e.g., **/dev/rdsk/c0t4d0s0** or **/dev/rmt/0mb**).

volname is the physical volume name; it should match the external sticker. Such label names are limited to six or fewer characters. The argument *volname* can be **-** to use the existing volume name.

The arguments *special1* and *volname1* are the device and volume, respectively, from which the copy of the file system is being extracted. The arguments *special2* and *volname2* are the target device and volume, respectively.

V

The **labelit** command can be used to provide initial labels for unmounted disk or tape file systems. With the optional arguments omitted, **labelit** prints current label values. **-F** and **-V** options can be specified for **labelit**. The behavior of **-F** and **-V** options are similar to the behavior of **volcopy**. The option **-n** of **labelit** provides for initial labeling of new tapes (this destroys previous contents).

WARNINGS

volcopy and **labelit** will be unsupported in a future release. Use *vxdump*(1M) and *vxrestore*(1M) for backup and restore, or use an application-specific utility. Use *dd*(1) to make a literal copy of the filesystem.

SEE ALSO

dd(1), labelit(1M), volcopy(1M), vxdump(1M), vxrestore(1M).

NAME

vtdaemon - respond to vt requests

SYNOPSIS

vtdaemon [**-g**[*ngateway*]] [**-n**] *lan_device lan_device* ...

DESCRIPTION

vtdaemon responds to requests from other systems (via local area network) made by **vt** (see *vt*(1)). **vtdaemon** spawns a server to respond to each request that it receives.

Options

vtdaemon recognizes the following command-line options and arguments:

-g[*ngateway*]
Causes **vtdaemon** to rebroadcast all requests received on one lan device to all other lan devices specified on the command line. The optional parameter *ngateway* specifies the maximum number of *vtgateway* servers that can be in operation concurrently. If *ngateway* is not specified, there is no limit on the number of vtgateway servers that can be in operation concurrently.

-n Causes *vtdaemon* to ignore all requests that have come through a gateway.

The remaining arguments are the full path names of lan devices that *vtdaemon* looks for requests on. If no lan devices are specified, the default lan device is used.

The major number for this device must correspond to a IEEE 802.3 local area network device.

Another function of **vtdaemon** is to create *portals* and service portal requests. A *portal* is a callout device that can be used by **uucico** to communicate to another machine via local area network (see *uucico*(1M)). Portals are created by **vtdaemon** according to the configuration information found in the file **/etc/uucp/L-vtdevices**. Each line in **L-vtdevices** has the format:

<calldev>[,<lan device>] <nodename>

For each line, **vtdaemon** creates a portal named *calldev* in **/dev**. Whenever this device is opened, **vtdaemon** spawns a server that creates a connection to the system specified by *nodename* via the lan device specified. If no lan device is specified, the first one specified on the command line when **vtdaemon** was started is used (or the default lan device is used if no lan devices were specified on the command line).

vtdaemon should be terminated by sending signal **SIGTERM** to it. When **vtdaemon** receives this signal it removes all of the portals it created in **/dev** before exiting.

DIAGNOSTICS

Diagnostics messages produced by **vtdaemon** are written to **/var/adm/vtdaemonlog**.

WARNINGS

vtdaemon uses the Hewlett-Packard **LLA** (Link Level Access) direct interface to the HP network drivers. **vtdaemon** uses the multicast address **0x01AABBCCBBAA**. It should not be used or deleted by other applications accessing the network. **vtdaemon** uses the following IEEE 802.3 *sap* (service access point) values: **0x90**, **0x94**, **0x98**, **0x9C**, **0xA0**, **0xA4**, **0xA8**, **0xAC**, **0xB0**, **0xB4**, **0xB8**, **0xBC**, **0xC0**, **0xC4**, **0xC8**, **0xCC**, **0xD0**, and **0xD4**. They should not be used by other applications accessing the network.

V

Desktop HP-UX

If your system has been installed with the Desktop HP-UX product, then both **ptydaemon** and **vtdaemon** will not be started by default. In order to start these daemons, change *PTYDAEMON_START* and *VTDAEMON_START* from a "0" to a "1" in the **/etc/rc.config.d/ptydaemon** and **/etc/rc.config.d/vt** files, respectively. The system must either be rebooted for these changes to take effect, or you can manually start both daemons by typing :

```
/usr/sbin/ptydaemon
/usr/sbin/vtdaemon /dev/lan0
```

where */dev/lan0* is the character special device file corresponding to the IEEE802.3 local area network device.

FILES

`/var/adm/vtdaemonlog`	logfile used by *vtdaemon*.
`/dev/lan0`	default lan device name.

SEE ALSO

vt(1), uucico(1M).

NAME

vxdiskusg - generate VxFS disk accounting data by user ID

SYNOPSIS

/usr/sbin/acct/vxdiskusg [**-s**] [**-v**] [**-i** *ignlist*] [**-p** *password-file*] [**-u** *outfile*] [*file*...]

DESCRIPTION

vxdiskusg generates intermediate disk accounting information from data in *file*, or the standard input if the **-s** option is specified and *file* is omitted. **vxdiskusg** outputs lines on the standard output, one line per user, in the following format:

uid login #blocks

where:

uid	User's numerical user ID
login	User's login name
#blocks	Total number of disk blocks allocated to this user

Without the **-s** option, *file* is the special filename of device(s) containing file systems. **vxdiskusg** reads only the inodes of file systems for disk accounting.

The output of **vxdiskusg** is normally the input to **acctdisk** (see *acct*(1M)) which generates total accounting records that can be merged with other accounting records. **vxdiskusg** is normally run in **dodisk** (see *acctsh*(1M)).

Options

vxdiskusg recognizes the following options:

-s Input data is already in **vxdiskusg** output format. **vxdiskusg** combines all lines for a single user into a single line.

-v verbose. Print a list on standard error of all files that are charged to no one.

-i *ignlist* Ignore the data on those file systems whose file system name is in *ignlist*. *ignlist* is a list of file system names, separated by commas or separated by spaces and enclosed within quotes. **vxdiskusg** compares each name in this list with the file system name stored in the *volume name* (see *volcopy_vxfs*(1M)), if it exists.

-p *password-file*
Use *password-file* as the name of the password file to generate login names. **/etc/passwd** is used by default.

-u *outfile* Write records to *outfile* of files that are charged to no one. Records consist of the special file name, the inode number, and the user ID.

EXAMPLES

The following generates daily disk accounting information for the file systems on these disks:

```
for i in /dev/vg00/lvol1 /dev/vg00/lvol6 /dev/vg00/lvol7; do
    vxdiskusg $i > dtmp.‘basename $i‘ &
done
wait
vxdiskusg -s dtmp.* | sort +0n +1 | acctdisk > disktacct
```

FILES

/etc/passwd used for converting user IDs to login names

SEE ALSO

acct(1M), acctsh(1M), volcopy(1M), volcopy_vxfs(1M), acct(4), diskusg(1M).

STANDARDS CONFORMANCE

vxdiskusg: SVID2, SVID3

NAME

vxdump, rvxdump - incremental VxFS file system dump, local or across network

SYNOPSIS

/usr/sbin/vxdump [**-nuwW**] [**-0123456789**] [**-f** *filename*] [**-d** *density*] [**-s** *size*] [**-T** *time*] [**-b** *blocksize*] [**-B** *records*] *filesystem*

/usr/sbin/rvxdump [**-nuwW**] [**-0123456789**] [**-f** *filename*] [**-d** *density*] [**-s** *size*] [**-T** *time*] [**-b** *blocksize*] [**-B** *records*] *filesystem*

/usr/sbin/vxdump *option* [*argument* ...] *filesystem*

/usr/sbin/rvxdump *option* [*argument* ...] *filesystem*

DESCRIPTION

vxdump copies to magnetic tape all files in the **vxfs** *filesystem* that have been changed after a certain date. This information is derived from the files **/var/adm/dumpdates** and **/etc/fstab**. **rvxdump** copies the files to a tape drive on a remote system. **rvxdump** runs a process, **/usr/sbin/rmt**, on the remote machine to access the tape device.

vxdump and **rvxdump** support both *getopt*(3C) and traditional **dump** command line invocations as shown above. The original **dump** command line style is supported for compatibility with previous versions of **vxdump** and for synonymy with the existing **dump** program used for hfs file systems. For the traditional command line style, *option* consists of characters from the set **0123456789bBdfnsTuWw** without any intervening white space.

On most devices **vxdump** detects end-of-media and prompts you to change the media if there is insufficient space, so it is not necessary to specify the size of the device. However, if the dump will require multiple tapes and the tapes are to be read using an older version of **vxrestore**, or if the tape device handles end-of-media in a way that **vxdump** doesn't recognize, then you must specify the size of the device using the **-B** option or a combination of the **-d** and **-s** options.

Options

If no arguments are given, the options are assumed to be **-9u** and a default file system is dumped to the default tape.

-*number* *number* is a single digit in the range [0-9] and indicates the dump level. All files modified since the last date stored in the file **/var/adm/dumpdates** for the same file system at a lesser dump level will be dumped. Thus, the option **-0** dumps the entire file system. If no date is determined by the level, the beginning of (UNIX) time is assumed.

-B *records*
The number of logical records per volume. The **vxdump** logical record size is 1024 bytes. *records* can also be specified with a suffix to indicate a unit of measure other than 1024 bytes. You can append a **k** or **K**, **m** or **M**, or **g** or **G**, to the number to indicate that the value is in kilobytes, megabytes, or gigabytes, respectively. This option overrides the calculation of tape size based on length and density.

-b *blocksize*
The blocking factor is taken from the *blocksize* option argument. (default is 63 if **-b** is not specified). Block size is defined as the logical record size times the blocking factor. **vxdump** writes logical records of 1024 bytes. Older versions of **vxdump** used a blocking factor of 10 for tapes with densities less than 6250 BPI, and 32 for tapes with densities of 6250 BPI or greater. **vxrestore** dynamically determines the blocking factor.

-d *density*
The density of the tape (expressed in BPI) used to calculate the amount of tape used per tape reel. If **-s** is specified, a default density value of 1600 is assumed for a reel tape.

-f *filename*
Place the dump on the file *filename* instead of the tape. If the name of the file is **-**(dash), **vxdump** writes to the standard output. This option can be of the form *machine***:***device* to specify a tape device on a remote machine.

-n Whenever **vxdump** requires operator attention, notify all users in group **operator** by means similar to that described by *wall*(1M).

V

-s *size* *size* is the size of the dump tape, specified in feet. When the specified size is reached, **vxdump** waits for reels to be changed. If **-d** is specified, a default size value of 2300 is assumed for a reel tape.

-T *date* Use the specified date as the starting time for the dump instead of the time determined from looking in **/var/adm/dumpdates**. The format of *date* is the same as that of *ctime*(3C) This option is useful for automated dump scripts that wish to dump over a specific period of time.

- You can specify **-T** only for incremental dumps; using **-T** for a level 0 dump returns an error.
- **-T** is mutually exclusive with the **-u** option.
- If you enter an improperly formatted date, **-T** returns an error message and terminates the dump.

-u If the dump completes successfully, write on file **/var/adm/dumpdates** the date when the dump started. This file records a separate date for each file system and each dump level. The format of **/var/adm/dumpdates** is user-readable and consists of one free-format record per line: file system name, increment level and dump date in *ctime*(3C) format. The file **/var/adm/dumpdates** can be edited to change any of the fields if necessary.

-W For each file system in **/var/adm/dumpdates**, print the most recent dump date and level, indicating which file systems should be dumped. If **-W** is specified, all other options are ignored and **vxdump** exits immediately.

-w Operate like **-W**, but print only file systems that need to be dumped.

Operator Interaction

vxdump requires operator intervention for any of the following conditions:

- end of tape
- end of dump
- tape-write error
- tape-open error
- disk-read error (if errors exceed threshold of 32).

In addition to alerting all operators implied by the **-n** option, **vxdump** interacts with the control terminal operator by posing questions requiring **yes** or **no** answers when it can no longer proceed or if there is a serious problem.

Because making a full dump typically requires considerable time, **vxdump** establishes a checkpoint at the start of each tape volume. If, for any reason, writing that volume fails, **vxdump**, with operator permission, restarts from the checkpoint after the old tape is rewound and removed and a new tape is mounted.

vxdump periodically reports information to the operator, including estimates (typically low) of the number of blocks to write, the number of tapes it requires, time required to complete, and the time remaining until tape change. The output is verbose to inform other users that the terminal controlling **vxdump** is busy and will be for some time.

Compatibility

The dump tape format is independent of the VxFS disk layout. A dump of a file system with the Version 4 disk layout can be restored on a file system using the Version 2 disk layout or even a file system of another file system type, with the following exceptions:

- Files larger than 2 GB cannot be restored by earlier versions of **vxrestore**. If a file larger than 2 GB is encountered, an older **vxrestore** skips the file and returns this message:

  Resync restore, skipped *num* **blocks**

- Files larger than 2 GB cannot be restored on a file system that does not support large files (see *mount_vxfs*(1M)).
- A file with a large *uid* (user ID of the file owner) or large *gid* (group ID of the file owner) cannot be restored correctly on a file system that does not support large IDs. Instead, the owner and/or group of the file will be that of the user invoking **vxrestore**. (A large ID is a value greater than 65535. The VxFS Version 2 disk layout does not support large IDs).

- Files with VxFS extent attributes (see *setext*(1M)) cannot be restored on a file system of a type that does not support extent attributes.

If you use **vxdump** to produce a dump intended for an earlier version of **vxrestore,** and if the dump requires multiple tapes, you should use the **-s**, **-d**, or **-B** option.

Dumps produced by older versions of **vxdump** can be read by the current version of **vxrestore**.

NOTES

Perform dumps with the file system unmounted or the system in single-user environment (see *init*(1M)) to ensure a consistent dump. If you have the HP OnLineJFS product installed, the dump can be performed in the multi-user environment using a snapshot file system with the online backup facility (see the **snapof=file** option of *mount_vxfs*(1M)).

Up to 32 read errors on the file system are ignored.

Each reel requires a new process; parent processes for reels already written remain until the entire tape is written.

vxdump does not dump information about ACLs, therefore **vxrestore** does not restore information about ACLs.

A version of **vxdump** resides in **/sbin** for use when the system is in single user state.

EXAMPLES

In the following example, assume that the file system **/mnt** is normally attached to the file tree at the root directory, (**/**).

In this example, the entire file system (**/mnt**) is dumped on **/dev/rmt/0m** and the size of the tape is 2 gigabytes.

```
vxdump -0 -B 2g -f /dev/rmt/0m /mnt
```

Using the traditional command line syntax and specifying the tape size in logical records:

```
vxdump 0Bf 2097152 /dev/rmt/0m /mnt
```

The option argument **2097152** goes with the option letter **B** as it is the first option letter that requires an option argument. The option argument **/dev/rmt/0m** goes with the option letter **f** as it is the second option letter that requires an option argument.

AUTHOR

vxdump and **rvxdump** are based on the **dump** and **rdump** programs from the 4.4 Berkeley Software Distribution, developed by the the University of California, Berkeley, and its contributors.

FILES

/dev/rmt/0m	Default tape unit to dump to.
/var/adm/dumpdates	New format-dump-date record.
/etc/fstab	Dump table: file systems and frequency.
/etc/mnttab	Mounted file system table.
/etc/group	Used to find group **operator**.

SEE ALSO

close(2), ctime(3C), dump(1M), fstab(4), getopt(3C), init(1M), mount_vxfs(1M), open(2), rmt(1M), setext(1M), vxrestore(1M), wall(1M).

NAME

vxenablef - enable VxFS DMAPI or OnLineJFS functionality in the kernel

SYNOPSIS

/sbin/fs/vxfs/vxenablef [**-a**] [**-e** { **online** | **dmapi** }]

DESCRIPTION

vxenablef enables VxFS DMAPI (Data Management Applications Programming Interface) or OnLineJFS functionality in the kernel. If you have valid licenses for these features, **vxenablef** makes them available without rebuilding a new kernel and rebooting the system.

If no argument is specified, **vxenablef** displays the available licensed features enabled in the kernel.

Options

vxenablef recognizes the following options.

-a Enable all features for which the proper license key exists.

-e online | **dmapi**
Enable OnLineJFS or DMAPI VxFS features.

online enables OnLineJFS functionality in the kernel if the system is licensed for OnLineJFS. The HP OnLineJFS product includes DMAPI.

dmapi enables DMAPI functionality in the kernel if the system is licensed for DMAPI. DMAPI is also part of the HP OnLineJFS product, but this option enables DMAPI without OnLineJFS.

NOTES

vxenablef is run automatically from **/etc/inittab** at system boot time. It is also run during the installation of the HP OnLineJFS product to enable the newly-licensed OnLineJFS functions without requiring a system reboot. It is not typically run from the command line.

You must be a privileged user to run **vxenablef**.

SEE ALSO

vxlicense(1M).

NAME

vxfsconvert - convert a file system to a vxfs file system

SYNOPSIS

/sbin/fs/vxfs/vxfsconvert [**-l** *logsize*] [**-s** *size*] [**-efnNvyY**] *special*

DESCRIPTION

vxfsconvert converts a file system of a supported type to a **vxfs** file system with a Version 4 disk layout. Currently **vxfsconvert** only supports conversion of an **hfs** file system to a **vxfs** file system. Conversion of all file system block and fragment sizes is supported.

Do a full backup of the file system before running **vxfsconvert**. File system conversion is complex and while most file systems will convert without problems, some may not. You could lose data if you don't have a backup. See the WARNINGS section.

vxfsconvert requires sufficient disk space to convert existing metadata to vxfs metadata. The space is acquired from free space within the file system or from the space available immediately after the end of the file system. In either case, the space must be available on the same device or volume that contains the file system. **vxfsconvert** requires approximately 12%-15% of the total file system size as free space, depending on the number of directories and files.

special is the character disk or volume manager device. Running **vxfsconvert** on the character device is usually faster than running it on a block device.

vxfsconvert converts HFS access control list (ACL) entries to the respective VxFS ACL entries with limitations. Only the entries that comply with the VxFS ACL standard are converted. See the description of the conversion process, below, for details.

vxfsconvert takes approximately 2 to 3 times longer to convert a file system than running a full **fsck** on an **hfs** file system.

Options

-e Estimate the amount of space required to complete the conversion. This option does not convert the file system to VxFS. No data is written to the file system and the file system remains clean.

-e generally overestimates the free space because it considers the worst case scenario for allocating blocks (that is, fully fragmented).

-f Display the list of supported file system types. Currently only **hfs**.

-l *logsize*
Specify the size of the file system intent log. The default *logsize* is the value of the **hfs** fragment size before conversion.

-n|**N** Assume a **no** response to all questions asked by **vxfsconvert**. This option implies that the conversion is never committed and the file system is not converted to VxFS.

-s *size* Specify the amount of available disk space at the end of the file system in kilobytes With this option all disk space required for the conversion process is taken from the end of the file system; the existing free space within the file system remains intact.

-v Specify verbose mode. Verbose mode shows the progress of the conversion process. For every inode converted, one of the following characters is displayed.

-	The inode is a regular file.
b	The inode is a block special file.
c	The inode is a character special file.
d	The inode is a directory.
l	The inode is a symbolic link.
p	The inode is a fifo.
s	The inode is a socket.
?	The inode type is unknown.

V

-y | Y Assume a **yes** response to all questions asked by **vxfsconvert**. This option implies that the conversion is committed unless **vxfsconvert** fails to allocate the required disk space. If an unknown inode type is detected during the conversion, **vxfsconvert** ignores it.

Conversion Process

To prepare a file system for conversion:

- Unmount the filesystem and make sure it is clean (you may need to use *fsck*(1M) to clean the filesystem). **vxfsconvert** cannot convert a mounted or dirty file system.
- Do a full backup on the file system before starting the conversion process.

Now run **vxfsconvert**. **vxfsconvert** does the following steps to convert a file system:

1. Examines the superblock to make sure it is marked CLEAN.
2. Based on information in the file system superblock, sets up VxFS metadata. This includes initializing all metadata required by the VxFS Version 4 disk layout (for example OLT, log, structural fileset). At this time, the original file system superblock is marked DIRTY unless you specified the **-e** or **-s** option.
3. Reads every inode in the file system and converts it to a VxFS inode.
4. For every regular file inode, **vxfsconvert** allocates and initializes enough extent data to map all of the file's data blocks. This translates only the representation of the file's data blocks from the old format to that of VxFS. It never copies or relocates user data blocks.
5. For every directory inode, **vxfsconvert** allocates sufficient disk space to hold all the directory entries. For every directory entry in that directory, **vxfsconvert** converts it to a VxFS directory entry and writes all converted directory blocks.
6. Converts all symbolic link, character special, block special, fifo, and socket inodes to VxFS.
7. Converts HFS ACL entries to the respective VxFS ACL entries. Only the entries that comply with the VxFS ACL standard are converted. The compliant entries are those that specify permissions for either a user or a group, but not both. That is, entries of format **(***user***.%)** and **(%.***group***)** will be converted, while entries of format **(***user.group***)** will be omitted. For files with both supported and unsupported entries all supported entries will be converted, but unsupported entries will be omitted.

Up to this point, all metadata of the original file system is intact and the conversion process can be stopped. The file system can be used after you run the original file system-specific **fsck**. If you specified the **-e** or **-s** option, running the file system-specific **fsck** is not required.

8. If all above steps completed successfully **vxfsconvert** asks whether to commit the conversion. It waits for the user response unless the **-y** or **-n** option was specified.
9. **vxfsconvert** replaces the original superblock with the VxFS superblock and clears any alternate superblocks written by the original file system. The VxFS superblock is never written if you have specified the **-n** or **-e** option. After the superblock is overwritten, the original file system is no longer accessible; it is now a VxFS file system.

Now you must run the VxFS-specific full **fsck** on the converted file system. During pass 4, **fsck** displays several error messages that require a **yes** response to complete the conversion process. These errors occur because **vxfsconvert** does not create all metadata files; you must run **fsck** to complete the process. No error messages display during passes zero through three. The following is sample **fsck** output after successful conversion.

```
# fsck -F vxfs -y -o full /dev/vg01/rlvol5

superblock indicates that intent logging was disabled
cannot perform log replay
pass0 - checking structural files
pass1 - checking inode sanity and blocks
pass2 - checking directory linkage
pass3 - checking reference counts
pass4 - checking resource maps
fileset 1 au 0 imap incorrect - fix (ynq)y
fileset 1 au 0 iemap incorrect - fix (ynq)y
fileset 999 au 0 imap incorrect - fix (ynq)y
fileset 999 au 0 iemap incorrect - fix (ynq)y
corrupted CUT entries, clear? (ynq)y
```

```
au 0 emap incorrect - fix? (ynq)y
au 0 summary incorrect - fix? (ynq)y
au 1 emap incorrect - fix? (ynq)y
au 1 summary incorrect - fix? (ynq)y
au 1 state file incorrect - fix? (ynq)y
fileset 1 iau 0 summary incorrect - fix? (ynq)y
fileset 999 iau 0 summary incorrect - fix? (ynq)y
free block count incorrect 0 expected 48878 fix? (ynq)y
free extent vector incorrect fix? (ynq)y
OK to clear log? (ynq)y
set state to CLEAN? (ynq)y
```

EXAMPLES

The following example checks available free space in the **/dev/vg01/lvol5** file system, unmounts the file system, and returns the amount of free space required for conversion. Available free space must always be greater than or equal to the required free space.

```
# df -k /dev/vg01/lvol5
/usr          (/dev/vg01/lvol5) :    43264 total allocated Kb
                                     30785 free allocated Kb
                                     12479 used allocated Kb
                                        28 % allocation used

# umount /dev/vg01/lvol5
# /sbin/fs/vxfs/vxfsconvert -e /dev/vg01/rlvol5
vxfs vxfsconvert: Total of 1219K bytes required to complete the conversion
```

To convert the file system, enter:

```
# /sbin/fs/vxfs/vxfsconvert /dev/vg01/rlvol5
vxfs vxfsconvert: Do you wish to commit to conversion? (ynq) y
vxfs vxfsconvert: CONVERSION WAS SUCCESSFUL
```

Upon successful conversion, check file system sanity, mount, and reorganize the file system (with **fsadm**) as in the following example:

```
# fsck -F vxfs -y -o full /dev/vg01/rlvol5
super-block indicates that intent logging was disabled
cannot perform log replay
pass0 - checking structural files
pass1 - checking inode sanity and blocks
pass2 - checking directory linkage
fileset 999 directory 3 block 7591 offset 952 has reclen 0 clear block? (ynq)y
pass3 - checking reference counts
pass4 - checking resource maps
fileset 1 au 0 imap incorrect - fix (ynq)y
fileset 1 au 0 iemap incorrect - fix (ynq)y
fileset 999 au 0 imap incorrect - fix (ynq)y
fileset 999 au 0 iemap incorrect - fix (ynq)y
fileset 999 au 0 imap incorrect - fix (ynq)y
fileset 999 au 0 iemap incorrect - fix (ynq)y
corrupted CUT entries, clear? (ynq)y
au 0 emap incorrect - fix? (ynq)y
au 0 summary incorrect - fix? (ynq)y
au 1 state file incorrect - fix? (ynq)y
au 1 emap incorrect - fix? (ynq)y
au 1 summary incorrect - fix? (ynq)y
au 1 state file incorrect - fix? (ynq)y
fileset 1 iau 0 summary incorrect - fix? (ynq)y
fileset 999 iau 0 summary incorrect - fix? (ynq)y
fileset 999 iau 0 summary incorrect - fix? (ynq)y
free block count incorrect 0 expected 35764 fix? (ynq)y
free extent vector incorrect fix? (ynq)y
OK to clear log? (ynq)y
set state to CLEAN? (ynq)y
```

V

```
# mount -F vxfs /dev/vg01/lvol5 /usr
# fsadm -ed /mntpt
fsadm: /etc/default/fs is used for determining the file system type
```

If the conversion fails, due to I/O failure, for example, run **fsck** to return to the original file system.

```
fsck -F hfs /dev/vg01/rlvol5
```

To convert a file system on a volume manager (for example, LVM) volume, using only disk space at the end of the file system (with the **-s** option), you may need to increase the volume size to provide the additional space to do the conversion. If LVM is used, you can do the following:

```
vxfsconvert -e /dev/vg01/rlvol5
lvextend -L new_size /dev/vg01/rlvol5
vxfsconvert -s required_space /dev/vg01/rlvol5
```

After the conversion completes, the increased volume space becomes a part of the converted VxFS file system.

Note: DO NOT reduce the volume after the conversion.

If the conversion fails, continue using the original file system. You do not need to run **fsck**. Reclaim the disk space by entering (on LVM):

```
lvreduce /dev/vg01/rlvol5 old_size
```

NOTES

To take full advantage of the VxFS file system, use **fsadm** to reorganize the file system after the conversion. The online reorganization feature of **fsadm** is available in only with the HP OnLineJFS product.

The converted VxFS file system uses the Version 4 disk layout, which is only recognized by JFS 3.3 and later versions. Do not convert a file system which is shared by a system running HP-UX 10.x or HP-UX 11.00 without JFS 3.3.

You must call **vxfsconvert** with its full pathname: **/sbin/fs/vxfs/vxfsconvert**.

Quota conversion is not supported.

WARNINGS

Do not run **vxfsconvert** on the **/stand** or whole-disk bootable file systems.

In the rare case of unsuccessful conversion, there is a risk of data loss or corruption. Always do a full system backup before the conversion.

DIAGNOSTICS

All error messages, I/O failure, and exit messages display on standard output.

SEE ALSO

fsck(1M), fsck_vxfs(1M), fsck_hfs(1M), mkfs_vxfs(1M), fsadm_vxfs(1M), lvm(7).

NAME

vxlicense - VxFS and VxVM licensing key utility

SYNOPSIS

/sbin/vxlicense {**-p** | **-t** *feature*}

DESCRIPTION

vxlicense maintains the VxFS and VxVM license key files.

Options

vxlicense requires one of the following options.

-p Print available VxFS and VxVM licenses and features in a system.

-t *feature* Test a VxFS and VxVM license in a system. *feature* can be **HP_OnLineJFS**, **HP_DMAPI**, or **VxVM**.

EXAMPLES

To list features available on your system, type:

```
vxlicense -p

vrts:vxlicense: INFO: Feature name: HP_OnLineJFS [50]
vrts:vxlicense: INFO: Number of licenses: 1 (non-floating)
vrts:vxlicense: INFO: Expiration date: Sun Sep 10 04:00:00 2000
vrts:vxlicense: INFO: Release Level: 22
vrts:vxlicense: INFO: Machine Class: All

vrts:vxlicense: INFO: Feature name: VxVM [95]
vrts:vxlicense: INFO: Number of licenses: 1 (non-floating)
vrts:vxlicense: INFO: Expiration date: Sun Sep 10 04:00:00 2000
vrts:vxlicense: INFO: Release Level: 31
vrts:vxlicense: INFO: Machine Class: All
```

NOTES

vxlicense is run automatically when you install the HP OnLineJFS product.

FILES

/etc/vx/elm/*.lic license file

SEE ALSO

vxenablef(1M).

NAME

vxrestore, rvxrestore - restore file system incrementally, local or across network

SYNOPSIS

/usr/sbin/vxrestore [**-himrRtvxy**] [**-b** *blocksize*] [**-e** *opt*] [**-f** *file*] [**-s** *number*] [*filename* ...]

/usr/sbin/rvxrestore [**-himrRtvxy**] [**-b** *blocksize*] [**-e** *opt*] [**-f** *file*] [**-s** *number*] [*filename* ...]

/usr/sbin/vxrestore *key* [*filename* ...]

/usr/sbin/rvxrestore *key* [*filename* ...]

DESCRIPTION

vxrestore and **rvxrestore** read tapes previously dumped by the **vxdump** or **rvxdump** command (see *vxdump*(1M)). **vxrestore** restores from tape on the local system; **rvxrestore** restores from tape on a remote system. **rvxrestore** runs **/usr/sbin/rmt** on the remote machine to access the tape device.

vxrestore and **rvxrestore** support both *getopt*(3C) and traditional **restore** command line invocations as shown above. The original **restore** command line style is supported for compatibility with previous versions of **vxrestore** and for synonymy with the existing **restore** program used for hfs file systems.

For the original **restore** command line style, actions taken are controlled by the *key* argument where *key* is a string of characters containing exactly one function letter from the group **irRtx**, and zero or more function modifiers from the group **befhmsvy**. One or more *filename* arguments, if present, are file or directory names specifying the files to restore. Unless the **h** modifier is specified (see below), the appearance of a directory name refers to the files and (recursively) subdirectories of that directory.

/dev/rmt/0m is the default tape device.

Options

-i Allow interactive restoration of files from a dump tape. After reading the directory information from the tape, **vxrestore** provides a shell-like interface that lets you move around the directory tree selecting files to extract. The available commands are listed below. For commands that require an argument, the default is the current directory.

add [*arg*]	Add the current directory or specified argument to the list of files to extract. If a directory is specified, the directory and all its descendents are added to the extraction list (unless the **h** key is specified on the command line). File names on the extraction list are displayed with a leading * when listed by **ls**.
cd [*arg*]	Change the current working directory to the specified argument.
delete [*arg*]	Delete the current directory or specified argument from the list of files to extract. If a directory is specified, the directory and all its descendents are deleted from the extraction list (unless **h** is specified on the command line). The best way to extract most files from a directory is to add the directory to the extraction list, then delete unnecessary files.
extract	Extract all files named on the extraction list from the dump tape. **vxrestore** prompts for the volume to mount. The fastest way to extract a few files is to start with the last volume, then work toward the first volume.
help	List a summary of the available commands.
ls [*arg*]	List the current or specified directory. Entries that are directories are displayed with a trailing **/**. Entries marked for extraction are displayed with a leading *. If the verbose key is specified, the inode number of each entry is also listed.
pwd	Print the full pathname of the current working directory.
quit	**vxrestore** immediately exits, even if the extraction list is not empty. **ctl-D** (control-D) is a synonym for **quit**.
set-modes	Set the owner, modes, and times of all directories that are added to the extraction list. Nothing is extracted from the tape. This setting is useful for cleaning

V

up after a restore aborts prematurely.

verbose The sense of the **v** modifier is toggled. When set to verbose, the **ls** command lists the inode numbers of all entries. and **vxrestore** prints information about each file as it is extracted.

-r Read the tape and load into the current directory. Be careful when using the **-r** option. Restore only a complete dump tape onto a clear file system, or restore an incremental dump tape after a full level zero restore. The following is a typical sequence to restore a complete dump:

```
/usr/sbin/newfs -F vxfs /dev/rdsk/c0t0d0
/usr/sbin/mount -F vxfs /dev/dsk/c0t0d0 /mnt
cd /mnt
vxrestore -r
```

You can then execute another **vxrestore** to restore an incremental dump on top of this. Note that **vxrestore** leaves a file, **restoresymtab**, in the root directory of the file system to pass information between incremental **vxrestore** passes. Remove this file when the last incremental tape is restored.

-R Resume a full restore. **vxrestore** restarts from a checkpoint it created during a full restore (see **-r** above). It requests a particular tape of a multi-volume set on which to restart a full restore. This provides a means for interrupting and restarting a multi-volume **vxrestore**.

-s *number*
number is the dump file number to recover. This is useful if there is more than one dump file on a tape.

-t Names of *filenames*, as specified on the command line, are listed if they occur on the tape. If no *filename* is given, the root directory is listed, which results in the entire content of the tape being listed, unless **-h** is specified.

-x Extract named files from the tape. If the named file matches a directory whose contents are written onto the tape, and the **-h** option is not specified, the directory is recursively extracted. The owner, modification time, and mode are restored (if possible). If no *filename* argument is given, the root directory is extracted, which results in the entire contents of the tape being extracted, unless **-h** is specified.

The following options can be used in addition to the letter that selects the primary function:

-b *blocksize*
Specify the block size of the tape in kilobytes. If the **-b** option is not specified, **vxrestore** determines the tape block size dynamically. [This option exists to preserve backwards compatibility with previous versions of **vxrestore**.]

-e *opt*
Specify how to handle a *vxfs* file that has extent attribute information. Extent attributes include reserved space, a fixed extent size, and extent alignment. It may not be possible to preserve the information if the destination file system does not support extent attributes, has a different block size than the source file system, or lacks free extents appropriate to satisfy the extent attribute requirements. Valid values for *opt* are:

force	Fail to restore the file if extent attribute information cannot be kept.
ignore	Ignore extent attribute information entirely.
warn	Issue a warning message if extent attribute information cannot be kept (the default).

-f *file*
Specify the name of the archive instead of **/dev/rmt/0m**. If the name of the file is **-**(dash), **vxrestore** reads from standard input. So you can use **vxdump** and **vxrestore** in a pipeline to vxdump and vxrestore a file system with the command

```
vxdump 0f - /usr | (cd /mnt; vxrestore xf -)
```

You can use an archive name of the form *machine***:***device* to specify a tape device on a remote machine.

-h Extract the actual directory, rather than the files to which it refers. This prevents hierarchical restoration of complete subtrees.

-m Extract by inode numbers rather than by file name. This is useful if only a few files are being extracted and you want to avoid regenerating the complete pathname to the file.

-v Specify verbose output; list the name of each file restored, preceded by its file type.

-y Do not ask whether to abort the operation if **vxrestore** encounters a tape error, but continue. Normally **vxrestore** asks whether to continue after encountering a read error. With this option, **vxrestore** continues without asking, skipping over the bad tape block(s) and continuing as best it can.

DIAGNOSTICS

vxrestore complains if a read error is encountered. If the **-y** option has been specified, or you respond **y**, **vxrestore** tries to continue the restore.

If the dump extends over more than one tape, **vxrestore** asks the user to change tapes. If the **-x** or **-i** option has been specified, **vxrestore** also asks which volume to mount. The fastest way to extract a few files is to start with the last volume and work towards the first volume.

There are numerous consistency checks that **vxrestore** can list. Most checks are self-explanatory or rarely occur. Here are some common errors:

filename: **not found on tape**
The specified file name was listed in the tape directory but not found on the tape. This is caused by tape read errors while looking for the file, and from using a dump tape created on an active file system.

expected next file *inumber*, **got** *inumber*
A file not listed in the directory appeared. This can occur when using a dump tape created on an active file system. Dumps should be performed with the file system unmounted or the system in single-user mode (see *init*(1M)) to insure a consistent dump. If the HP OnLineJFS product is installed, the dump can be performed in the multi-user environment using a snapshot file system with the online backup facility (see the **snapof=file** option of *mount_vxfs*(1M)).

Incremental tape too low
When doing an incremental restore, a tape that was written before the previous incremental tape, or that has too low an incremental level was loaded.

NOTE: if this error occurs, you are either restoring tapes out of order or restoring from a dump file that was created using the **-T** option to **vxdump**. At this point, **vxrestore** displays a warning message and asks if you want to continue doing the restore. Respond with **y** only if you are sure that you are restoring from a dump file created using the **-T** option. Enter **n** to abort the restore.

Incremental tape too high
When doing an incremental restore, a tape that does not begin its coverage where the previous incremental tape left off, or that has too high an incremental level was loaded.

NOTE: If this error occurs, you are either restoring tapes out of order or restoring from a dump file that was created using the **-T** option to **vxdump**. At this point **vxrestore** displays a warning message and asks if you want to continue doing the restore. Respond with **y** only if you are sure that you are restoring from a dump file created using the **-T** option. Enter **n** to abort the restore.

Tape read error while restoring *filename*
Tape read error while skipping over inode *inumber*
Tape read error while trying to resynchronize
A tape-read error occurred. If a file name is specified, the contents of the restored files may be incorrect. If **vxrestore** is skipping an inode or is trying to resynchronize the tape, no extracted files are corrupted, although files may not be found on the tape.

Resync restore, skipped *num* **blocks**
After a tape-read error, **vxrestore** may have to resynchronize itself. This message indicates the number of blocks skipped over. This message will also be generated by older versions of **vxrestore** while skipping over files larger than 2 gigabytes dumped by a more recent version of **vxdump**.

NOTES

If the dump tape contains files larger than 2 gigabytes, and if the file system being restored to does not support files larger than 2 gigabytes, the file is not restored correctly. Instead it is truncated to 2 gigabytes.

A file with a large *uid* (user ID of the file owner) or large *gid* (group ID of the file owner) cannot be restored correctly on a file system that does not support large IDs. Instead, the owner and/or group of the file will be that of the user invoking **vxrestore**. (A large ID is a value greater than 65535. The VxFS Version 2 disk layout does not support large IDs).

The current version of **vxrestore** can read dumps produced by older versions of **vxdump**.

vxrestore can restore files to a file system of a type other than VxFS. If the file system type does not support extent attributes, than the extent attributes are not restored (see the **-e** option).

A version of **vxrestore** resides in **/sbin** for use when the system is in single user state.

WARNINGS

vxrestore can get confused when doing incremental restores from dump tapes that were made on active file systems.

A level 0 dump (see the *vxdump*(1M) manual page) must be done after a full restore. Because **vxrestore** runs in user code, it has no control over inode allocation; thus a full dump must be done to get a new set of directories reflecting the new inode numbering, even though the contents of the files are unchanged.

vxrestore does not restore access control lists (ACLs).

AUTHOR

vxrestore and **rvxrestore** are based on the **restore** program distributed in the 4.4 Berkeley Software Distribution, developed by the the University of California, Berkeley, and its contributors.

FILES

/dev/rmt/0m	default tape drive
/tmp/rstdr*	file containing directories on the tape
/tmp/rstmd*	owner, mode, and time stamps for directories
./restoresymtab	information passed between incremental restores

SEE ALSO

extendfs_vxfs(1M), fsadm_vxfs(1M), getopt(3C), init(1M), mkfs_vxfs(1M), mount_vxfs(1M), newfs_vxfs(1M), restore(1M), rmt(1M), vxdump(1M).

NAME

vxtunefs - tune a VxFS File System

SYNOPSIS

/sbin/vxtunefs [**-ps**] [**-f** *tunefstab*] [**-o** *parameter*=*value*] [{*mount_point* | *block_special*}]...

DESCRIPTION

vxtunefs sets or prints tuneable I/O parameters of mounted file systems. **vxtunefs** can set parameters describing the I/O properties of the underlying device, parameters to indicate when to treat an I/O as direct I/O, or parameters to control the extent allocation policy for the specified file system.

With no options specified, **vxtunefs** prints the existing VxFS parameters for the specified file systems.

vxtunefs works on a list of mount points specified on the command line, or all the mounted file systems listed in the **tunefstab** file. The default **tunefstab** file is **/etc/vx/tunefstab**. You can change the default using the **-f** option.

vxtunefs can be run at any time on a mounted file system, and all parameter changes take immediate effect. Parameters specified on the command line override parameters listed in the **tunefstab** file.

If **/etc/vx/tunefstab** exists, the VxFS-specific **mount** command invokes **vxtunefs** to set device parameters from **/etc/vx/tunefstab**.

Options

-f *filename*
Use *filename* instead of **/etc/vx/tunefstab** as the file containing tuning parameters.

-o *parameter*=*value*
Specify parameters for the file systems listed on the command line. The parameters are listed below.

-p Print the tuning parameters for all the file systems specified on the command line.

-s Set the new tuning parameters for the VxFS file systems specified on the command line or in the **tunefstab** file.

VxFS Tuning Parameters and Guidelines

The values for all the following parameters except **read_nstream** and **write_nstream** can be specified in bytes, kilobytes, megabytes or sectors (512 bytes) by appending **k**, **K**, **m**, **M**, **s**, or **S**. There is no need for a suffix for the value in bytes.

For an application to do efficient direct I/O or discovered direct I/O, it should issue read requests that are equal to the product of **read_nstream** and **read_pref_io**. In general, any multiple or factor of **read_nstream** multiplied by **read_pref_io** is a good size for performance. For writing, the same general rule applies to the **write_pref_io** and **write_nstream** parameters. When tuning a file system, the best thing to do is use the tuning parameters under a real workload.

If an application is doing sequential I/O to large files, it should issue requests larger than the **discovered_direct_iosz**. This performs the I/O requests as discovered direct I/O requests which are unbuffered like direct I/O, but which do not require synchronous inode updates when extending the file. If the file is too large to fit in the cache, using unbuffered I/O avoids losing useful data out of the cache, and lowers CPU overhead.

The VxFS tuneable parameters are:

default_indir_size

On VxFS, files can have up to 10 variable sized extents stored in the inode. After these extents are used, the file must use indirect extents which are a fixed size that is set when the file first uses indirect extents. These indirect extents are 8K by default. The file system does not use larger indirect extents because it must fail a write and return ENOSPC if there are no extents available that are the indirect extent size. For file systems with many large files, the 8K indirect extent size is too small. The files that get into indirect extents use a lot of smaller extents instead of a few larger ones. By using this parameter, the default indirect extent size can be increased so that large files in indirects use fewer larger extents.

Be careful using this tuneable. If it is too large, then writes fail when they are unable to allocate extents of the indirect extent size to a file. In general, the fewer and the larger the files on a file system, the larger **default_indir_size** can be. The value of this parameter is generally a multiple of the **read_pref_io** parameter.

This tuneable does not apply to disk layout Version 4.

discovered_direct_iosz
Any file I/O requests larger than the **discovered_direct_iosz** are handled as discovered direct I/O. A discovered direct I/O is unbuffered like direct I/O, but it does not require a synchronous commit of the inode when the file is extended or blocks are allocated. For larger I/O requests, the CPU time for copying the data into the buffer cache and the cost of using memory to buffer the I/O becomes more expensive than the cost of doing the disk I/O. For these I/O requests, using discovered direct I/O is more efficient than regular I/O. The default value of this parameter is 256K.

initial_extent_size
Changes the default size of the initial extent.

VxFS determines, based on the first write to a new file, the size of the first extent to allocate to the file. Typically the first extent is the smallest power of 2 that is larger than the size of the first write. If that power of 2 is less than 8K, the first extent allocated is 8K. After the initial extent, the file system increases the size of subsequent extents (see **max_seqio_extent_size**) with each allocation.

Because most applications write to files using a buffer size of 8K or less, the increasing extents start doubling from a small initial extent. **initial_extent_size** changes the default initial extent size to a larger value, so the doubling policy starts from a much larger initial size, and the file system won't allocate a set of small extents at the start of file.

Use this parameter only on file systems that have a very large average file size. On such file systems, there are fewer extents per file and less fragmentation.

initial_extent_size is measured in file system blocks.

max_buf_data_size
Determines the maximum buffer size allocated for file data. The two accepted values are 8K bytes and 64K bytes. The larger value can be beneficial for workloads where large reads/writes are performed sequentially. The smaller value is preferable on workloads where the I/O is random or is done in small chunks. The default value is 8K bytes.

max_direct_iosz
Maximum size of a direct I/O request issued by the file system. If there is a larger I/O request, it is broken up into **max_direct_iosz** chunks. This parameter defines how much memory an I/O request can lock at once; do not set it to more than 20% of memory.

max_diskq
Limits the maximum disk queue generated by a single file. When the file system is flushing data for a file and the number of pages being flushed exceeds **max_diskq**, processes block until the amount of data being flushed decreases. Although this does not limit the actual disk queue, it prevents synchronizing processes from making the system unresponsive. The default value is 1 megabyte.

max_seqio_extent_size
Increases or decreases the maximum size of an extent. When the file system is following its default allocation policy for sequential writes to a file, it allocates an initial extent that is large enough for the first write to the file. When additional extents are allocated, they are progressively larger (the algorithm tries to double the size of the file with each new extent), so each extent can hold several writes worth of data. This reduces the total number of extents in anticipation of continued sequential writes. When there are no more writes to the file, unused space is freed for other files to use.

In general, this allocation stops increasing the size of extents at 2048 blocks, which prevents one file from holding too much unused space.

max_seqio_extent_size is measured in file system blocks.

read_nstream
The number of parallel read requests of size **read_pref_io** to have outstanding at one time. The file system uses the product of **read_nstream** and **read_pref_io** to determine its read ahead size. The default value for **read_nstream** is 1.

read_pref_io
The preferred read request size. The file system uses this in conjunction with the **read_nstream** value to determine how much data to read ahead. The default value is 64K.

`write_nstream`
The number of parallel write requests of size **`write_pref_io`** to have outstanding at one time. The file system uses the product of **`write_nstream`** and **`write_pref_io`** to determine when to do flush behind on writes. The default value for **`write_nstream`** is 1.

`write_pref_io`
The preferred write request size. The file system uses this in conjunction with the **`write_nstream`** value to determine how to do flush behind on writes. The default value is 64K.

FILES

`/etc/vx/tunefstab` VxFS file system tuning parameters table.

SEE ALSO

mount_vxfs(1M), mkfs_vxfs(1M), tunefstab(4).

NAME

vxupgrade - upgrade the disk layout of a VxFS file system

SYNOPSIS

/usr/sbin/vxupgrade [**-n** *new_version*] [**-r** *rawdev*] *mount_point*

DESCRIPTION

vxupgrade prints the current disk layout version number for a VxFS file system or upgrades the file system to a new disk layout. **vxupgrade** operates on file systems mounted for read/write access: *mount_point* must be a mounted VxFS file system. Only a privileged user can query or upgrade a VxFS file system.

When invoked with the **-n** option, **vxupgrade** upgrades the disk layout to the specified version. When invoked without the **-n** option, **vxupgrade** prints the disk layout version number of the file system.

To perform an upgrade, **vxupgrade** freezes the file system, allocates and initializes the new structures, frees the space used by the old structures, and then thaws the file system. This process should not keep the file system frozen for more than a few seconds.

vxupgrade employs a lock file (**lost+found/.fsadm**) on the file system to ensure that only one instance of **vxupgrade** is running at any time. **vxupgrade** and **fsadm** cannot run simultaneously, so the lock file also ensures that **vxupgrade** does not run while a file system reorganization is in progress. When **vxupgrade** is invoked for an upgrade, it opens the lock file in the root of the file system specified by *mount_point*. If the lock file doesn't exist, it is created. The *fcntl*(2) system call is used to obtain a write lock on the file. If the write lock fails, **vxupgrade** fails, assuming that another **vxupgrade** or an **fsadm** is running.

Options

-n *new_version*	Upgrade disk layout to *new_version*. *new_version* can be 3 or 4.
-r *rawdev*	Use the pathname *rawdev* as the raw device. This option can be used when **vxupgrade** cannot determine which raw device corresponds to the mount point (when **/etc/mnttab** is corrupted, for example).

Free Space Requirement

vxupgrade requires free space on the file system to perform the upgrade; the upgrade may fail if there is not enough free space. It is difficult to determine the exact amount of space required to upgrade a VxFS file system, however, you can estimate the maximum space required.

To upgrade a disk layout Version 2 file system with $n * 1024$ inodes (allocated only) and $m * 32768$ blocks to disk layout Version 3, the worst-case minimum value is at least $n * 2432$ bytes + $m * 8220$ bytes + 115 kilobytes, in extents of 8 kilobytes or larger. Free extents of larger than 8 kilobytes may be required, so this is only a lower bound on the worst-case minimum required. Since this is the worst-case minimum, it may be possible to upgrade with less free space available. After an upgrade to a new disk layout is completed, all of this free space, plus some additional free space, will be reclaimed.

You cannot upgrade a Version 2 disk layout to Version 4 directly. You must first upgrade from Version 2 to a Version 3 disk layout, and then upgrade to Version 4. The upgrade may fail due to a lack of space at each step.

NOTES

Once a file system has been upgraded to Version 3, it is no longer mountable on HP-UX 10.01 and 10.10.

Once a file system has been upgraded to Version 4, it is no longer mountable on:

- HP-UX 10.*x*
- HP-UX 11.0 without JFS 3.3 from Application CD

Version 4 file systems are mountable on:

- HP-UX 11.0 with JFS 3.3 from Application CD
- HP-UX 11.1*x*

You cannot upgrade the root (**/**) or **/usr** file systems to Version 4 on an 11.0 system running JFS 3.3 from the Application CD. Additionally, we do not advise upgrading the **/var** or **/opt** file systems to Version 4 on an 11.0 system. These core file systems are crucial for system recovery. The HP-UX 11.0 kernel and emergency recovery media were built with an older version of JFS that does not recognize the Version 4 disk layout. If these file systems were upgraded to Version 4, your system would fail to boot with the 11.0 kernel as delivered or the emergency recovery media. You can, however, upgrade these core file systems to Version 4 on an HP-UX 11.1*x* system.

Disk layout versions cannot be downgraded, for example, you cannot change a file system from disk layout version 4 to disk layout version 3.

A file system cannot be upgraded from a Version 3 disk layout to a Version 4 disk layout if its intent log size is less than 256 kilobytes.

After upgrading from a Version 2 disk layout, run **fsadm -c** *mount-point* to convert the inode format to allow growth beyond a two-gigabyte offset.

DIAGNOSTICS

vxupgrade returns an exit value of 0 if the upgrade was successful, 1 if the upgrade failed due to a lack of free space, and 2 if the upgrade failed for some other reason.

FILES

mount_point/**lost+found/.fsadm**
lock file

SEE ALSO

fsadm_vxfs(1M), mkfs_vxfs(1M), vxfsio(7).

NAME

wall - write message to all users

SYNOPSIS

/usr/sbin/wall [**-g***groupname*] [*file*]

/usr/sbin/cwall [**-g***groupname*] [*file*]

DESCRIPTION

Without arguments, the **wall** command reads a message from standard input until end-of-file. Then it sends this message to all currently logged-in users preceded by:

Broadcast Message from ...

If the **-g***groupname* option is specified, **wall** sends the message to all currently logged-in *groupname* members (as specified in **/etc/group**) preceded by:

Broadcast Message from ... **to group** *groupname*

If *file* is specified, **wall** reads *file* instead of standard input.

wall is typically used to warn all users prior to shutting down the system.

The sender must have appropriate privileges to override any protections the users may have invoked (see *mesg*(1)).

wall has timing delays, and takes at least 30 seconds to complete.

You must have appropriate privileges to override any protections users may have invoked (see *mesg*(1)).

EXTERNAL INFLUENCES

International Code Set Support

Single- and multibyte character code sets are supported.

DIAGNOSTICS

Cannot send to ...

The open on a user's tty file failed.

WARNINGS

The **wall** command will be WITHDRAWN from X/Open standard and may not be portable to other vendor's platforms.

AUTHOR

wall was developed by AT&T and HP.

FILES

/dev/tty*

SEE ALSO

mesg(1), write(1).

STANDARDS CONFORMANCE

wall: SVID2, SVID3, XPG2, XPG3

NAME

whodo - which users are doing what

SYNOPSIS

/usr/sbin/whodo [**-h**] [**-l**] [**user**]

DESCRIPTION

The **whodo** command produces merged, reformatted, and dated output from the **who**, **ps** and **acctcom** commands (see *who*(1) , *ps*(1) and *acctcom(1M))*.

If user is specified, output is restricted to all sessions pertaining to that user.

The following options are available:

-h Suppress the heading.

-l Produce a long form of output. The fields displayed are: the user's login name, the name of the tty the user is on, the time of day the user logged in (in hours:minutes), the idle time - that is, the time since the user last typed anything (in hours:minutes), the CPU time used by all processes and their children on that terminal (in minutes:seconds), the CPU time used by the currently active processes (in minutes:seconds), and the name and arguments of the current process.

EXTERNAL INFLUENCES

Environment Variables

LC_COLLATE determines the order in which the output is sorted.

If **LC_COLLATE** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**. If any internationalization variable contains an invalid setting, **whodo** behaves as if all internationalization variables are set to "C" (see *environ*(5)).

FILES

/etc/passwd

/var/adm/pacct

SEE ALSO

ps(1), who(1), acctcom(1M).

STANDARDS CONFORMANCE

whodo: SVID2, SVID3

NAME

xntpd - Network Time Protocol daemon

SYNOPSIS

xntpd [**-abdm**] [**-c** *conffile*] [**-e** *authdelay*] [**-f** *driftfile*]
[**-k** *keyfile*] [**-l** *logfile*] [**-p** *pidfile*] [**-r** *broadcastdelay*]
[**-s** *statsdir*] [**-t** *key*] [**-v** *variable*] [**-V** *variable*] [**-x**]

DESCRIPTION

xntpd is an operating system daemon which sets and maintains the system time-of-day in synchronism with Internet standard time servers. **xntpd** is a complete implementation of the Network Time Protocol (NTP) version 3, as defined by RFC-1305, but also retains compatibility with version 1 and 2 servers as defined by RFC-1059 and RFC-1119, respectively.

xntpd does all computations in 64-bit fixed point arithmetic and requires no floating point support. While the ultimate precision of this design, about 232 picoseconds, is not achievable with ordinary workstations and networks of today, it may be required with future nanosecond CPU clocks and gigabit LANs.

The daemon can operate in any of several modes, including symmetric active/passive, client/server and broadcast/multicast, as described in RFC-1305. A broadcast/multicast client can discover remote servers, compute server-client propagation delay correction factors and configure itself automatically. This makes it possible to deploy a group of workstations without specifying configuration details specific to the local environment.

Ordinarily, **xntpd** reads the **/etc/ntp.conf** configuration file at startup time in order to determine the synchronization sources and operating modes. It is also possible to specify a working, although limited, configuration entirely on the command line, obviating the need for a configuration file. This may be particularly appropriate when the local host is to be configured as a broadcast or multicast client, with all peers being determined by listening to broadcasts at run time. Various internal **xntpd** variables can be displayed and configuration options altered while the daemon is running using the **ntpq** and **xntpdc** utility programs.

COMMAND LINE OPTIONS

-a	Enable authentication mode. The default is disable.
-b	Synchronize using NTP broadcast messages.
-c *conffile*	Specify the name and path of the configuration file.
-d	Specify debugging mode. This flag may occur multiple times, with each occurrence indicating greater detail of display.
-e *authdelay*	Specify the time (in seconds) it takes to compute the NTP encryption field on this computer.
-f *driftfile*	Specify the name and path of the drift file.
-k *keyfile*	Specify the name and path of the file containing the NTP authentication keys.
-l *logfile*	Specify the name and path of the log file. The default is the system log facility.
-p *pidfile*	Specify the name and path to record the daemon's process ID.
-r *broadcastdelay*	Specify the default propagation delay from the broadcast/multicast server and this computer. This is used only if the delay cannot be computed automatically by the protocol.
-s *statsdir*	Specify the directory path for files created by the statistics facility.
-t *key*	Add a key number to the trusted key list.
-v *variable*	Add a system variable.
-V *variable*	Add a system variable listed by default.
-x	Make all adjustments by SLEW.

THE CONFIGURATION FILE

The **xntpd** configuration file is read at initial startup in order to specify the synchronization sources, modes and other related information. Usually, it is installed in the **/etc/ntp.conf** directory, but could be installed elsewhere (see the **-c conffile** command line option).

The file format is similar to other UNIX configuration files. Comments begin with a # character and extend to the end of the line. Blank lines are ignored. Configuration commands consist of an initial keyword followed by a list of arguments, some of which may be optional, separated by white space. Commands may not be continued over multiple lines. Arguments may be host names, host addresses written in numeric, dotted-quad form, integers, floating point numbers (when specifying times in seconds) and text strings. Optional arguments are delimited by **[ ]** in the following descriptions, while alternatives are separated by **|**. The notation **[...]** means an optional, indefinite repetition of the last item before the **[...]**.

While there is a rich set of options available, the only required option is one or more server, peer or broadcast commands described in the "Configuration Options" section. The examples in **/etc/ntp.conf.example** may also be helpful.

CONFIGURATION OPTIONS

The configuration commands are as decribed below:

peer *address* [**key** *key_id*] [**version** *version_id*] [**prefer**]

server *address* [**key** *key_id*] [**version** *version_id*] [**prefer**] [**mode** *mode*]

broadcast *address* [**key** *key_id*] [**version** *version_id*] [**ttl** *ttl*]

The above three commands can be used to specify either the name or address of the time server and the mode in which the time server should operate. The address can be either a DNS name or an IP address in dotted-quad notation.

The **peer** command specifies that the local server is to operate in symmetric active mode with the remote server. In this mode, the local server can be synchronized to the remote server and, in addition, the remote server can be synchronized by the local server. This is useful in a network of servers where, depending on various failure scenarios, either the local or remote server may be the better source of time.

The **server** command specifies that the local server is to operate in client mode with the specified remote server. In this mode, the local server can be synchronized to the remote server, but the remote server can never be synchronized to the local server. This is the most common operating mode (by far).

The **broadcast** command specifies that the local server is to operate in broadcast mode, where the local server sends periodic broadcast messages to a client population at the broadcast/multicast address specified. Ordinarily, this specification applies only to the local server operating as a sender; for operation as a broadcast client, see the **broadcastclient** or **multicastclient** commands below. In this mode, address is usually the broadcast address of (one of) the local network(s) or a multicast address assigned to NTP. The address of 224.0.1.1 is assigned to NTP. This is presently the only address that should be used. Note that the use of multicast features requires a multicast kernel.

OPTIONS

key *key_id*
All packets sent to the address are to include authentication fields encrypted using the specified key identifier, which is an unsigned 32 bit integer. The default is to not include an encryption field.

version *version_id*
Specifies the version number to be used for outgoing NTP packets. Versions 1, 2, and 3 are the choices, with version 3 the default.

prefer
Marks the server as preferred. All other things being equal, this host will be chosen for synchronization among a set of correctly operating hosts.

ttl *ttl*
This option is used only with broadcast mode. It specifies the ttl (time-to-live) to use on multicast packets. Selection of the proper value, which defaults to 127, must be co-ordinated with the network administrator(s).

broadcastclient [*address*]
This command directs the local server to listen for broadcast messages at the broadcast address of the local network. The default address is the subnet address with the host field bits set to ones. Upon hearing a broadcast message for the first time, the local server measures the nominal network delay using a brief client/server exchange with the remote server, then enters the broadcastclient mode, in which it listens for and synchronizes to succeeding broadcast messages. Note that, in order to avoid accidental or malicious disruption in this mode, both the local and remote servers should operate using authentication and the same trusted key and key identifier.

driftfile *driftfile*
This command specifies the name of the file used to record the frequency offset of the local

X

clock oscillator. If the file exists, it is read at startup in order to set the initial frequency offset and then updated once per hour with the current frequency offset computed by the daemon. If the file does not exist or this command is not given, the initial frequency offset is assumed zero. In this case, it may take some hours for the frequency to stabilize and the residual timing errors to subside.

enable *auth* | *bclient* | *monitor* | *pll* | *pps* | *stats*

disable *auth* | *bclient* | *monitor* | *pll* | *pps* | *stats*

Provides a way to enable or disable various server options. Flags not mentioned are unaffected. Note that all of these flags can be controlled remotely using the **xntpdc** utility program. Each of these flags are described below.

auth Enables the server to synchronize with unconfigured peers only if the peer has been correctly authenticated using a trusted key and key identifier. The default for this flag is disable.

bclient
Enables the server to listen for a message from a broadcast or multicast server, as in the multicastclient command with default address. The default for this flag is disable.

monitor
Enables the monitoring facility. See the **xntpdc monlist** command for further information (*xntpdc*(1M)).

pll Enables the server to adjust its local clock, with default enable. If not set, the local clock free-runs at its intrinsic time and frequency offset. This flag is useful in case the local clock is controlled by some other device or protocol and NTP is used only to provide synchronization to other clients. In this case, the local clock driver is used. See the **Reference Clock Drivers** for further information.

pps Enables the pulse-per-second (PPS) signal when frequency and time is disciplined by the precision time kernel modifications. The default is enable when these modifications are available and disable otherwise.

stats Enables the statistics facility. By default this option is enabled.

Authentication Key File Format

The NTP standard specifies an extension allowing verification of the authenticity of received NTP packets, and to provide an indication of authenticity in outgoing packets. This is implemented in **xntpd** using the DES encryption algorithm. The specification allows any one of a possible 4 billion keys, numbered with 32 bit unsigned integers, to be used to authenticate an association. The servers involved in an association must agree on the value of the key used to authenticate their data, though they must each learn the key independently. The keys are standard 56 bit DES keys.

Additionally, another authentication algorithm is available which uses an MD5 message digest to compute an authenticator. The length of the key or password is limited to 8 characters. **xntpd** reads its keys from a file specified using the **-k** command line option or the **keys** statement in the configuration file. While key number 0 is fixed by the NTP standard (as 56 zero bits) and may not be changed, one or more of the keys numbered 1 through 15 may be arbitrarily set in the keys file.

The key file uses the same comment conventions as the configuration file. Key entries use a fixed format of the form

keyno type key

where *keyno* is a positive integer, *type* is a single character which defines the format the key is given in, and *key* is the key itself.

X

The key may be given in one of four different formats, controlled by the *type* character. The four key types, and corresponding formats, are listed following.

S The *key* is a 64 bit hexadecimal number in the format specified in the DES document, that is the high order 7 bits of each octet are used to form the 56 bit key while the low order bit of each octet is given a value such that odd parity is maintained for the octet. Leading zeroes must be specified (i.e. the key must be exactly 16 hex digits long) and odd parity must be maintained. Hence a zero key, in standard format, would be given as *0101010101010101* .

N The *key* is a 64 bit hexadecimal number in the format specified in the NTP standard. This is the same as the DES format except the bits in each octet have been rotated one bit right so that the parity bit is now the high order bit of the octet. Leading zeroes must be specified and odd parity must be maintained. A zero key in NTP format would be specified as *8080808080808080*

A The *key* is a 1-to-8 character ASCII string. A key is formed from this by using the lower order 7 bits of the ASCII representation of each character in the string, with zeroes being added on the right when necessary to form a full width 56 bit key, in the same way that encryption keys are formed from Unix passwords.

M The *key* is a 1-to-32 character ASCII string, using the MD5 authentication scheme. Note that both the keys and the authentication schemes (DES or MD5) must be identical between a set of peers sharing the same key number.

FILEGEN UTILITY

filegen *name* [**file** *filename*] [**type** *typename*] [**link** | **nolink**] [**enable** | **disable**]

This command helps in configuring the generation-file set **name**. The generation-file sets provides a mean for handling files that are continuously growing during the lifetime of a server. Server statistics are a typical example for such files. The generation-file sets provide access to a set of files used to store the actual data. At any time at most one element of the set is being written to. The **type** given specifies when and how data will be directed to a new element of the set. This way, information stored in elements of a file set that are currently unused are available for administrational operations without the risk of disturbing the operation of **xntpd**. (Most important: they can be removed to free space for new data produced.) Filenames of set members are built from three elements. *name* is name of the statistic to be collected. Currently only two kinds of statistics are supported: *loopstats* and *peerstats*.

file Defines a *filename* string directly concatenated to the *prefix* mentioned above (no intervening / (slash)) if *prefix* is defined in the **statsdir** statement.

type This part reflects individual elements of a file set. It is generated according to the *type* of a file set as explained below. A file generation set is characterized by its type. The following *typenames* are supported:

none The file set is actually a single plain file.

pid One element of file set is used per incarnation of a **xntpd** server. This type does not perform any changes to file set members during runtime, however it provides an easy way of separating files belonging to different **xntpd** server incarnations. The set member filename is built by appending a dot **.** to the concatenated *prefix* and *filename* strings, and appending the decimal representation of the process id of the **xntpd** server process. (e.g *<prefix><filename>.<pid>*)

day One file generation set element is created per day. The term *day* is based on *UTC*. A day is defined as the period between 00:00 and 24:00 UTC. The file set member suffix consists of a dot and a day specification in the form *<YYYYMMDD>*. *YYYY* is a 4 digit year number (e.g. 1992). *MM* is a two digit month number. *DD* is a two digit day number. Thus, all information written at December 10th, 1992 would end up in a file named *<prefix><filename>.19921210*

week Any file set member contains data related to a certain week of a year. The term *week* is defined by computing day of year modulo 7. Elements of such a file generation set are distinguished by appending the following suffix to the file set filename base: A dot, a four digit year number, the letter **W** and a two digit week number. For example, information from January, 10th 1992 would end up in a file with suffix *.1992W1*.

month One generation file set element is generated per month. The file name suffix consists of a dot, a four digit year number, and a two digit month.

year One generation file elment is generated per year. The filename suffix consists of a dot and a 4 digit year number.

age This type of file generation sets changes to a new element of the file set every 24 hours of server operation. The filename suffix consists of a dot, the letter **a**, and an eight digit number. This number is taken to be the number of seconds the server is running at the start of the corresponding 24 hour period.

enabled/disabled
Information is only written to a file generation set when this set is **enabled**. Output is

prevented by specifying **disabled**. The default is **enabled**.

link/nolink

It is convenient to be able to access the *current* element of a file generation set by a fixed name. This feature is enabled by specifying **link** and disabled using **nolink**. The default is **link**. If **link** is specified, a hard link from the current file set element to a file without suffix is created. When there is already a file with this name and the number of links of this file is one, it is renamed appending a dot, the letter **C**, and the pid of the **xntpd** server process. When the number of links is greater than one, the file is unlinked. This allows the current file to be accessed by a constant name.

REFERENCE CLOCK DRIVERS

Individual clocks can be activated by configuration file commands, specifically the **server** and **fudge** commands described in the *xntpdc*(1M) manual page. The following discussion presents information on how to select and configure the device drivers in a running UNIX system.

Radio and modem clocks by convention have addresses in the form **127.127.t.u**, where **t** is the clock type and **u** is a unit number in the range 0-3 used to distinguish multiple instances of clocks of the same type. Most of these clocks require support in the form of a serial port or special bus peripheral. The particular device is normally specified by adding a soft link **/dev/deviceu** to the particular hardware device involved, where **u** correspond to the unit number above.

Following is a list showing the type and title of each driver currently implemented. The compile-time identifier for each is shown in parentheses.

The following four clock drivers are supported by HP.

Type 1 Local Clock Driver (LOCAL_CLOCK)
Type 4 Spectracom 8170 and Netclock/2 WWVB Receivers (WWVB)
Type 26 Hewlett Packard 58503A GPS Receiver (HPGPS)
Type 29 Trimble Palisade GPS Receiver (PALISADE)

The clock drivers mentioned below are not supported by HP. They may work, but have not been tested. They are provided **as is**, for the convenience of the diverse users.

Type 2 Trak 8820 GPS Receiver (TRAK)
Type 3 PSTI/Traconex 1020 WWV/WWVH Receiver (PST)
Type 5 TrueTime GPS/GOES/OMEGA Receivers (TRUETIME)
Type 8 Generic Reference Driver (PARSE)
Type 10 Austron 2200A/2201A GPS Receivers (AS2201)
Type 11 * TrueTime OMEGA Receiver
Type 15 * TrueTime TM-TMD GPS Receiver
Type 16 Bancomm GPS/IRIG Receiver (HP only) (BANC)
Type 17 Datum Precision Time System (DATUM)
Type 18 NIST Modem Time Service (ACTS)
Type 20 Generic NMEA GPS Receiver (NMEA)
Type 23 PTB Modem Time Service (PTBACTS)
Type 24 USNO Modem Time Service (USNO)
Type 25 * TrueTime generic.

All TrueTime receivers are now supported by one driver, type 5. Types 11, 15 and 25 will be retained only for a limited time and may be reassigned in future.

DEBUGGING HINTS FOR REFERENCE CLOCK DRIVERS

The **ntpq** and **xntpdc** utility programs can be used to debug reference clocks, either on the server itself or from another machine elsewhere in the network. The server is compiled, installed and started using the command-line switches described in the *xntpdc*(1M) manual page.

X

The first thing to look for are error messages on the system log. If none occur, the daemon has started, opened the devices specified and waiting for peers and radios to come up.

The next step is to be sure the RS232 messages, if used, are getting to and from the clock. The most reliable way to do this is with an RS232 tester and to look for data flashes as the driver polls the clock and/or as data arrive from the clock. Our experience is that many of the problems occurring during installation are due to problems such as miswired connectors or improperly configured device links at this stage.

If RS232 messages are getting to and from the clock, variables can be inspected using the **ntpq** command (see *ntpq*(1M)). First, use the **pe** and **as** commands to display billboards showing the peer configuration and association IDs for all peers, including the radio clock peers. The assigned clock address should appear in the **pe** billboard and the association ID for it at the same relative line position in the **as** billboard. If things are operating correctly, after a minute or two samples should show up in the **pe** display line for the clock.

Additional information is available with the **rv** and **clockvar** commands, which take as argument the association ID shown in the **as** billboard. The **rv** command with no argument shows the system variables, while the **rv** command with association ID argument shows the peer variables for the clock, as well as any other peers of interest. The **clockvar** command with argument shows the peer variables specific to reference clock peers, including the clock status, device name, last received timecode (if relevant), and various event counters. In addition, a subset of the fudge parameters is included.

The **xntpdc** utility program can be used for detailed inspection of the clock driver status. The most useful are the **clockstat** file and the commands in **xntpdc** (see*xntpdc*(1M)).

Most drivers write a message to the clockstats file as each timecode or surrogate is received from the radio clock. By convention, this is the last ASCII timecode (or ASCII gloss of a binary-coded one) received from the radio clock. This file is managed by the **filegen** facility described above and requires specific commands in the configuration file. This forms a highly useful record to discover anomalies during regular operation of the clock.

SLEW

SLEW is not recommended by HP because it reduces accuracy and stability.

The **NTP** daemon has three regimes in which it operates:

offset below 128 milliseconds

This is the normal operating regime of **NTP**. A properly configured **NTP** hierarchy (with reasonable networking) can operate for years without ever approaching the 128 millisecond upper limit. All time adjustments are small and smooth (known as SLEWING), and nobody even notices the **SLEW** adjustments unless they have a cesium clock or a **GPS** clock and expensive instrumentation to make sophisticated measurements (HP/Agilent makes the instruments).

offset above 128 milliseconds

This regime is often encountered at power-on because, those battery-backed real-time clocks they put in computers are not too great. Because **NTP** is quite capable of keeping the offset below one millisecond all the time it is running, many users want to get into the normal regime quickly when an offset above 128 millisecond is encountered at startup. So in this situation **NTP** will (fairly quickly) make a single STEP change, and is usually successful in getting the offset well below 128 millisecond so there will be no more of the disruptive STEP changes.

offset above 1000 seconds

This is so far out of the normal operating range that **NTP** decides something is terribly wrong and human intervention is required. The daemon shuts down.

The catch is that the dispersion on a WAN is frequently much greater than 128 milliseconds, so you may see (a lot of) the STEP changes, perhaps as large as 1000 milliseconds (depends on your network). But there are customer applications that are quite allergic to the STEP changes, particularly backward steps (which will happen about half the time). Databases and financial transaction systems are examples.

The good news is that **NTP** can be forced to never make a STEP, but instead SLEW the clock to drive the offset to zero. This is accomplished with the **-x** option on the command line. This effectively removes the middle operating regime. You won't get millisecond (or microsecond) precision with this method, but you probably can't get that over a WAN anyway.

It is important to note that SLEWING is a cover-up for a more fundamental problem (poor connection to the timesource), and it does not solve this problem. SLEWING is not recommended by HP because it causes reduced accuracy and stability, and it leads to anamolous behavior that can be quite confusing. For example, you will see messages similar to this in the syslog:

```
: time reset (step) -411.093665 s
: synchronisation lost
: synchronized to 15.13.115.194, stratum=1
: Previous time adjustment incomplete; residual -0.022223 sec
: Previous time adjustment incomplete; residual -0.020624 sec
: Previous time adjustment incomplete; residual -0.020222 sec
: Previous time adjustment incomplete; residual -0.020623 sec
: Previous time adjustment incomplete; residual -0.020623 sec
```

But this does not mean that your system clock has been stepped. Only the **NTP** daemon process has seen a step in its notion of the current time (and this will be passed on to clients). The system time is being gradually adjusted in a series of **SLEW** maneuvers, and the **SLEW** rate is quite limited. Be warned that it can take a long time for the system clock to reach nominal correctness, and much longer to stabilize. Each cpu model is unique, but the maximum slew rate is typically about 40 milliseconds per second. Thus a **SLEW** adjustment of 411 seconds will take over 10,000 seconds (about 3 hours) to complete. A better approach would be to run the **ntpdate** command once at system startup, and accept the one STEP change that comes with it. Then start the **NTP** daemon process **xntpd** and it will never make a STEP as long as your connection to the timesource is good. This method also overcomes the 1000 seconds problem. The **NTP** startup script **/sbin/rc2.d/S660xntpd** will do this automatically if you configure the **NTPDATE_SERVER** variable in **/etc/rc.config.d/netdamons**. A properly configured **NTP** hierarchy with average networking (say 10Base-T) can run for years without ever making a STEP change.

AUTHOR

xntpd was developed by Dennis Ferguson at the University of Toronto.

Text amended by David Mills at the University of Delaware.

FILES

/etc/ntp.conf	The default configuration file
/etc/ntp.drift	The default drift file
/etc/ntp.keys	The default key file

SEE ALSO

ntpq(1M), ntpdate(1M), xntpdc(1M).

X

NAME

xntpdc - special NTP query program

SYNOPSIS

xntpdc [**-dilnps**] [**-c** *command*] [*host*] [**...**]

DESCRIPTION

xntpdc is used to query the **xntpd** daemon about its current state and to request changes in that state. The program may be run either in interactive mode or controlled mode using command line arguments. Extensive state and statistics information is available through the **xntpdc** interface. In addition, nearly all the configuration options which can be specified at start up using **xntpd**'s configuration file may also be specified at run time using **xntpdc**. If one or more request options is included on the command line when **xntpdc** is executed, each of the requests will be sent to the NTP servers running on each of the hosts given as command line arguments, or on localhost by default. If no request options are given, **xntpdc** will attempt to read commands from the standard input and execute these on the NTP server running on the first host given on the command line, again defaulting to localhost when no other host is specified. **xntpdc** will prompt for commands if the standard input is a terminal device.

xntpdc uses NTP mode 7 packets to communicate with the NTP server, and hence can be used to query any compatible server on the network which permits it. Note that since NTP is a UDP protocol, this communication will be somewhat unreliable, especially over large distances in terms of network topology. **xntpdc** makes no attempt to retransmit requests, and will timeout requests if the remote host is not heard from within a suitable timeout time.

The operation of **xntpdc** is specific to the particular implementation of the **xntpd** daemon and can be expected to work only with this and maybe some previous versions of the daemon. Requests from a remote **xntpdc** program which affect the state of the local server must be authenticated, which requires both the remote program and local server to share a common key and key identifier.

COMMAND LINE OPTIONS

Specifying a command line option other than **-i** or **-n** will cause the specified query (or queries) to be sent to the indicated host(s) immediately. Otherwise, **xntpdc** will attempt to read interactive format commands from the standard input.

-c *command* The following *command* is interpreted as an interactive format command and is added to the list of commands to be executed on the specified host(s). Multiple **-c** commands may be given.

-d Debugging information is printed.

-i Force **xntpdc** to operate in interactive mode. Prompts will be written to the standard output and commands read from the standard input.

-l Obtain a list of peers which are known to the server(s). This option is equivalent to **-c listpeers** command. See "CONTROL MESSAGE COMMANDS" below.

-n Output all host addresses in dotted-quad numeric format (xxx.xxx.xxx.xxx) rather than converting to the canonical host names.

-p Print a list of peers known to the server as well as a summary of their state. This is equivalent to **-c peers** command. See "CONTROL MESSAGE COMMANDS" below.

-s Print a list of peers known to the server as well as a summary of their state, but in a slightly different format than the **-p** command. This is equivalent to **-c dmpeers** command. See "CONTROL MESSAGE COMMANDS" below.

INTERACTIVE COMMANDS

Interactive format commands consist of a keyword followed by zero to four arguments. Only enough characters of the full keyword to uniquely identify the command need be typed. The output of a command is normally sent to the standard output. The output of individual commands may be redirected or sent to a file by appending a **>**, followed by a file name, to the command line.

A number of interactive format commands are executed entirely within the **xntpdc** program itself and do not result in NTP mode 7 requests being sent to a server. These commands are described as follows:

? [*command_keyword*]

help [*command_keyword*]
A **?** or **help** by itself will print a list of all the command keywords. **ntpq**. A **?** or **help** followed by a command keyword (*command_keyword*) will print function and usage information about the command.

delay *milliseconds*
Specify a time interval to be added to timestamps included in requests which require authentication. This is used to enable (unreliable) server reconfiguration over long delay network paths or between machines whose clocks are unsynchronized.

host *hostname* Set the host to which future queries will be sent. The *hostname* may be either a host name or a numeric address.

hostnames [**yes**|**no**]
If **yes** is specified, host names are printed in information displays. If **no** is specified, numeric addresses are printed instead. The default is **yes**, unless modified using the command line **-n** *command*.

keyid *keyid* This command allows the specification of a key number to be used to authenticate configuration requests. The *keyid* must correspond to a key number that the server has been configured to use for this purpose.

quit Exit **xntpdc**.

passwd This command prompts you to type in a password (which will not be echoed) which will be used to authenticate configuration requests. The password must correspond to the key configured for use by the NTP server for this purpose if such requests are to be successful.

timeout *milliseconds*
Specify a timeout period for responses to server queries. The default is about 8000 milliseconds. Note that since **xntpdc** retries each query once after a timeout, the total waiting time for a timeout will be twice the timeout value set.

CONTROL MESSAGE COMMANDS

Query commands result in NTP mode 7 packets containing requests for information being sent to the server. These are read-only commands in that they make no modification of the server configuration state.

listpeers Obtains and prints a brief list of the peers for which the server is maintaining state. This list should include all configured peer associations as well as those peers whose stratum is such that they are considered by the server to be possible future synchronization candidates.

peers Obtains a list of peers for which the server is maintaining state, along with a summary of that state. Summary information includes the address of the remote peer, the local interface address (0.0.0.0 if a local address has yet to be determined), the stratum of the remote peer (a stratum of 16 indicates the remote peer is unsynchronized), the polling interval in seconds, the reachability register in octal, and the current estimated delay, offset and dispersion of the peer, all in seconds. In addition, the character in the left margin indicates the mode this peer entry is operating in.

+	indicates symmetric active
-	indicates symmetric passive
=	indicates the remote server is being polled in client mode
^	indicates the server is broadcasting to this address
~	indicates the remote peer is sending broadcasts
*****	indicates the peer that the server is currently synchronizing to.

The contents of the host field may be a host name, an IP address, a reference clock implementation name with its parameter or REFCLK (implementation number, parameter). For **hostnames no**, only IP addresses will be displayed.

dmpeers
A slightly different peer summary list. The output is similar to that of the **peers** command, except for the character in the leftmost column. Characters only appear beside peers which were included in the final stage of the clock selection algorithm. A period (**.**) indicates that this peer

X

was cast off in the falseticker detection. A plus (**+**) indicates that the peer made it through. An asterisk (*****) denotes the peer that the server is currently synchronizing with.

showpeer *peer_address* [...]
Shows a detailed display of the current peer variables for one or more peers. Most of these values are described in the NTP Version 2 specification.

pstats *peer_address* [...]
Show per-peer statistic counters associated with the specified peer(s).

clockinfo *clock_peer_address* [...]
Obtain and print information concerning a peer clock. The values obtained provide information on the setting of fudge factors and other clock performance information.

kerninfo
Obtain and print kernel phase-lock loop operating parameters. This information is available only if the kernel has been specially modified for a precision timekeeping function.

loopinfo [**oneline**|**multiline**]
Print the values of selected loop filter variables. The loop filter is the part of NTP which deals with adjusting the local system clock. The offset is the last offset given to the loop filter by the packet processing code. The frequency is the frequency error of the local clock in parts-per-million (ppm). The time_const controls the stiffness of the phase-lock loop and thus the speed at which it can adapt to oscillator drift. The watchdog timer value is the number of seconds which have elapsed since the last sample offset was given to the loop filter. The **oneline** and **multiline** options specify the format in which this information is to be printed. **multiline** is the default.

sysinfo
Print a variety of system state variables, i.e., the state related to the local server.

The system flags show various system flags, some of which can be set and cleared by the **enable** and **disable** configuration commands, respectively. The configurable flags are the *auth*, *bclient*, *monitor*, *pll*, *pps* and *stats* flags. Refer to (*xntpd*(1M)) for the description of these flags.

The stability is the residual frequency error remaining after the system frequency correction is applied and is intended for maintenance and debugging. In most architectures, this value will initially decrease from as high as 500 ppm to a nominal value in the range .01 to 0.1 ppm. If it remains high for some time after starting the daemon, something may be wrong with the local clock, or the value of the kernel variable tick may be incorrect.

The **broadcastdelay** shows the default broadcast delay, as set by the **broadcastdelay** configuration command.

The **authdelay** shows the default authentication delay, as set by the **authdelay** configuration command.

sysstats
Print statistics counters maintained in the protocol module.

memstats
Print statistics counters related to memory allocation code.

iostats
Print statistics counters maintained in the input-output module.

timerstats
Print statistics counters maintained in the timer/event queue support code.

reslist
Obtain and print the server's restriction list. This list is (usually) printed in sorted order and may help to understand how the restrictions are applied.

monlist [*version*]
Obtain and print traffic counts collected and maintained by the monitor facility. The version number should not normally need to be specified.

clkbug *clock_peer_address*[*...*]
Obtain debugging information for a reference clock driver. This information is provided only by some clock drivers and is mostly undecodable without a copy of the driver source.

RUNTIME CONFIGURATION REQUESTS

All requests which cause state changes in the server are authenticated by the server using a configured NTP key. This facility is disabled if the NTP key is not configured. The key number and the corresponding key must also be made known to **xtnpdc**. This can be done using the keyid and passwd commands, the latter of which will prompt at the terminal for a password to use as the encryption key. You will also be prompted automatically for both the key number and password the first time a command which would result in an authenticated request to the server is given. Authentication not only provides verification that the requester has permission to make such changes, but also gives an extra degree of protection against transmission errors.

Authenticated requests always include a timestamp in the packet data, which is included in the computation of the authentication code. This timestamp is compared by the server to its receive time stamp. If they differ by more than a small amount the request is rejected. This is done for two reasons. First, it makes simple replay attacks on the server, by someone who might be able to overhear traffic on your LAN, much more difficult. Second, it makes it more difficult to request configuration changes to your server from topologically remote hosts. While the reconfiguration facility will work well with a server on the local host, and may work adequately between time-synchronized hosts on the same LAN, it will work very poorly for more distant hosts. As such, if reasonable passwords are chosen, care is taken in the distribution and protection of keys and appropriate source address restrictions are applied, the run time reconfiguration facility should provide an adequate level of security.

The following commands all make authenticated requests.

addpeer peer_address [*keyid*][*version*][*prefer*]
: Add a configured peer association at the given address and operating in symmetric active mode. Note that an existing association with the same peer may be deleted when this command is executed, or may simply be converted to conform to the new configuration, as appropriate. If the optional **keyid** is a nonzero integer, all outgoing packets to the remote server will have an authentication field (encrypted) attached with this key. If the value is 0 (or not given) no authentication will be done. The **version** # can be 1, 2 or 3 and defaults to 3. The **prefer** keyword indicates a preferred peer (and thus will be used primarily for clock synchronization if possible). The preferred peer also determines the validity of the PPS signal - if the preferred peer is suitable for synchronization so is the PPS signal.

addserver peer_address [*keyid*][*version*][*prefer*]
: Identical to the addpeer command, except that the operating mode is client.

broadcast peer_address [*keyid*][*version*][*prefer*]
: Identical to the **addpeer** command, except that the operating mode is broadcast. In this case a valid key identifier and key are required. The **peer_address** parameter can be the broadcast address of the local network or a multicast group address assigned to NTP. If using a multicast address, a multicast-capable kernel is required.

unconfig peer_address [...]
: This command causes the configured bit to be removed from the specified peer(s). In many cases this will cause the peer association to be deleted. When appropriate, however, the association may persist in an unconfigured mode if the remote peer is willing to continue on in this fashion.

fudge peer_address [*time1*][*time2*][*stratum*][*refid*]
: This command provides a way to set certain data for a reference clock. See the source listing for further information.

enable [*flag*][...]

disable [*flag*][...]
: These commands operate in the same way as the **enable** and **disable** configuration file commands of **xntpd**. Described below are the flags supported.

auth Enables the server to synchronize with unconfigured peers only if the peer has been correctly authenticated using a trusted key and key identifier. The default for this flag is enable.

bclient
: Enables the server to listen for a message from a broadcast or multicast server, as in the multicastclient command with default address. The default for this flag is disable.

X

monitor
Enables the monitoring facility. See the **xntpdc** program and the **monolist** command for more information. The default for this flag is enable.

pll Enables the server to adjust its local clock by means of NTP. If disabled, the local clock free-runs at its intrinsic time and frequency offset. This flag is useful in case the local clock is controlled by some other device or protocol and NTP is used only to provide synchronization to other clients. In this case, the local clock driver is used. The default for this flag is enable.

pps Enables the pulse-per-second (PPS) signal when frequency and time is disciplined by the precision time kernel modifications. The default for this flag is disable.

stats Enables the statistics facility. The default for this flag is enable.

restrict address mask flag [*flag*]
This command operates in the same way as the **restrict** configuration file commands of **xntpd**.

unrestrict address mask flag [*flag*]
Unrestrict the matching entry from the restrict list.

delrestrict address mask [*nttport*]
Delete the matching entry from the restrict list.

readkeys
Causes the current set of authentication keys to be purged and a new set to be obtained by rereading the keys file (which must have been specified in the **xntpd** configuration file). This allows encryption keys to be changed without restarting the server.

trustkey [*keyid*][...]

untrustkey [*keyid*][...]
These commands operate in the same way as the **trustedkey** and **untrustkey** configuration file commands of **xntpd**.

authinfo
Returns information concerning the authentication module, including known keys and counts of encryptions and decryptions which have been done.

traps Display the traps set in the server. See the source listing for further information.

addtrap [*address*][*port*][*interface*]
Set a trap for asynchronous messages. See the source listing for further information.

clrtrap [*address*][*port*][*interface*]
Clear a trap for asynchronous messages. See the source listing for further information.

reset Clear the statistics counters in various modules of the server. See the source listing for further information.

AUTHOR

xntpdc was developed by David L. Mills.

SEE ALSO

xntpd(1M), ntpdate(1M), ntpq(1M).

NAME

ypinit - build and install Network Information Service databases

SYNOPSIS

/usr/sbin/ypinit -m [**DOM=***NIS_domain*]

/usr/sbin/ypinit -s *NIS_server_name* [**DOM=***NIS_domain*]

Remarks

The Network Information Service (NIS) was formerly known as Yellow Pages (yp). Although the name has changed, the functionality of the service remains the same.

DESCRIPTION

ypinit is a shell script that creates Network Information Service (NIS) databases on either a master or slave NIS server. **ypinit** asks a few self-explanatory questions, and reports success or failure to the terminal. For an overview of Network Information Service, see *ypfiles*(4) and *ypserv*(1M).

Options

ypinit recognizes the following options and command-line arguments:

-m Create the local host as the master server to all maps (databases) provided in the NIS domain (see *domainname*(1)). All maps are built from scratch, either from information provided to **ypinit** at run-time, or from ASCII files in **/etc**. All such files should be complete and unabbreviated, unlike how they may exist on a NIS client machine (see *passwd*(4) for examples of abbreviated files).

See *ypmake*(1M) for more information on how NIS databases are built on the master server. Note that **ypinit** uses the **NOPUSH=1** option when invoking **make**, so newly formed maps are not immediately copied to slave servers (see *ypmake*(1M)).

-s Create NIS databases on a slave server by copying the databases from an existing NIS server that serves the NIS domain.

The *NIS_server_name* argument should be the host name of either the master server for all the maps or a server on which the maps are current and stable.

DOM=*NIS_domain* Causes **ypinit** to construct maps for the specified *NISdomain*. **DOM** defaults to the NIS domain shown by the **domainname** command (see *domainname*(1).

DIAGNOSTICS

ypinit returns exit code 0 if no errors occur; otherwise, it returns exit code 1.

AUTHOR

ypinit was developed by Sun Microsystems, Inc.

FILES

/etc/group
/etc/hosts
/etc/netgroup
/etc/networks
/etc/passwd
/etc/protocols
/etc/publickey
/etc/rpc
/etc/services
/etc/auto_master
/etc/mail/aliases

SEE ALSO

domainname(1), makedbm(1M), ypmake(1M), yppush(1M), ypserv(1M), ypxfr(1M), ypxfrd(1M), group(4), hosts(4), netgroup(4), networks(4), passwd(4), protocols(4), publickey(4), rpc(4), services(4), ypfiles(4).

NAME

ypmake - create or rebuild Network Information Service databases

SYNOPSIS

/var/yp/ypmake [**DIR=***source_directory*] **[**DOM=*NIS_domain*] \
[**NOPUSH=1**] [**PWFILE=***passwd_file*] [*map* [*map* ...]]

cd /var/yp; make [**DIR=***source_directory*] **[**DOM=*NIS_domain*] \
[**NOPUSH=1**] [**PWFILE=***passwd_file*] [*map* ...]

Remarks

The Network Information Service (NIS) was formerly known as Yellow Pages (yp). Although the name has changed, the functionality of the service remains the same.

DESCRIPTION

ypmake is a shell script that builds one or more Network Information Service (NIS) maps (databases) on a master NIS server. If no arguments are specified, **ypmake** either creates maps if they do not already exist or rebuilds maps that are not current. These maps are constructed from ASCII files. **ypmake** then executes **yppush** to notify slave NIS servers of the change and make the slave servers copy the updated maps to their machines (see *yppush*(1M)).

If any *maps* are supplied on the command line, **ypmake** creates or updates those *maps* only. Permissible names for *maps* are the filenames in **/etc** listed under FILES below. In addition, specific *maps* can be named, such as **netgroup.byuser** or **rpc.bynumber**.

The **make** command can be used instead of **ypmake** (see *make*(1)). The **Makefile** no longer calls the **ypmake** script but now actually constructs the maps. All NIS commands have been modified to use the **Makefile** instead of **ypmake**. The **Makefile** and **ypmake** can co-exist, but it is recommended that you consider using the **Makefile** which is the standard mechanism for building maps on other vendor's machines.

Both the **Makefile** and **ypmake** script use four variables:

DIR=*source_directory*	The directory containing the ASCII source files from which maps are constructed. **DIR** defaults to **/etc**.
DOM=*NIS_domain*	Causes **ypmake** to construct maps for the specified *NIS_domain*. **DOM** defaults to the NIS domain shown by **domainname** (see *domainname*(1)).
NOPUSH=1	When non-null (null by default), **NOPUSH** inhibits copying the new or updated databases to the slave NIS servers. Only slave NIS servers in the specified *domain* receive **yppush** notification when **NOPUSH** is null.
NOPUSH=2	Does the same thing as **NOPUSH=1** and sends a "clear current map" request to the local **ypserv** process.
PWFILE=*passwd_file*	Specifies the full pathname of the ASCII file that **ypmake** should use when building the NIS passwd maps. **PWFILE** defaults to **$DIR/passwd**.

The order of arguments passed to **ypmake** is unimportant, but the maps are built or updated in the left-to-right order provided.

Refer to *ypfiles*(4) and *ypserv*(1M) for an overview of Network Information Service.

DIAGNOSTICS

ypmake returns one of the following exit codes upon completion:

0	Normal termination; no problems.
1	One or more unrecognized arguments were passed.
2	The NIS domain name is not set.
3	The subdirectory used to contain maps for a specific NIS domain, **/var/yp/domain_name**, does not exist or is not writable.
4	An error was encountered when building at least one of the maps.
5	One or more maps' ASCII files do not exist or are unreadable.

EXAMPLES

Create or rebuild the password databases (both the **passwd.byname** and **passwd.byuid** maps) from **/etc/passwd** and use **yppush** to copy the databases to any slave NIS servers in the default NIS *domain*:

```
ypmake passwd.byname
```

Create or rebuild the hosts databases from **/etc/hosts** but do not copy the databases to any slave NIS servers:

```
ypmake hosts NOPUSH=1
```

Create or rebuild the network maps from **/nis/sourcefiles/networks** and copy the maps to any slave NIS servers in NIS domain **DAE_NIS**:

```
ypmake DOM=DAE_NIS networks DIR=/nis/sourcefiles
```

AUTHOR

ypmake was developed by Sun Microsystems, Inc.

FILES

/etc/group
/etc/hosts
/etc/netgroup
/etc/networks
/etc/passwd
/etc/protocols
/etc/publickey
/etc/rpc
/etc/services

SEE ALSO

domainname(1), make(1), makedbm(1M), ypinit(1M), yppush(1M), ypserv(1M), group(4), hosts(4), netgroup(4), networks(4), passwd(4), protocols(4), publickey(4), rpc(4), services(4), ypfiles(4).

NAME

yppasswdd - daemon for modifying Network Information Service passwd database

SYNOPSIS

/usr/lib/netsvc/yp/rpc.yppasswdd *passwd_file* [**-l** *log_file*] [**-nogecos**] [**-noshell**]
[**-nopw**] [**-m** [*arg1 arg2* ...]]

Remarks

The Network Information Service (NIS) was formerly known as Yellow Pages (yp). Although the name has changed, the functionality of the service remains the same.

DESCRIPTION

The **yppasswdd** daemon handles password change requests from **yppasswd** (see *yppasswd*(1)). It changes a password entry in *passwd_file*, which must be in the format defined by *passwd*(4). The change is made only if the old password provided by **yppasswd** matches the encrypted password of that entry.

yppasswdd should be executed only on the master Network Information Service (NIS) server for the passwd database (map). The **yppasswdd** daemon is not executed by default, nor can it be started by **inetd** (see *inetd*(1M)). To enable automatic startup of **yppasswdd** at boot time, the **NIS_MASTER_SERVER** variable should be set to 1 in file **/etc/rc.config.d/namesvrs** on the master NIS server.

Options

yppasswdd recognizes the following options and command-line arguments:

-l *log_file* Log diagnostic and error messages to *log_file*. These messages are not available if **yppasswdd** is started without the **-l** option.

Information logged to the file includes date and time of the message, the host name, process ID and name of the function generating the message, and the message itself. Note that different services can share a single log file because enough information is included to uniquely identify each message.

-nogecos -noshell -nopw
If these are given, then these fields may not be changed remotely using *passwd*(1).

-m [*arg1 arg2* ...] After *passwd_file* is modified, and if using the **-m** option, **yppasswdd** executes **make** to update the NIS passwd database (see *ypmake*(1M) Any arguments following the **-m** flag are passed to **make**.

To ensure that the passwd map is rebuilt to contain the new password and all slave NIS servers have their passwd maps properly updated to include the change, always use the **-m** option to **yppasswdd**, but do not use the **NOPUSH=1** argument to **make**.

EXAMPLES

Assume the **yppasswdd** daemon is started on the master NIS server as follows:

```
/usr/lib/netsvc/yp/rpc.yppasswdd /var/yp/src/passwd \
                              -l /var/adm/yppasswdd.log \
                              -m passwd DIR=/var/yp/src
```

This indicates that the ASCII file from which the NIS passwd database is built is **/var/yp/src/passwd**. When this file is updated by a request from **yppasswd**, the NIS passwd database is rebuilt and copied to all slave NIS servers in the master's NIS domain (see *domainname*(1)).

Log messages are written to the file **/var/adm/yppasswdd.log**.

WARNINGS

yppasswdd uses lock file **/var/adm/ptmp** to get exclusive access to *passwd_file* when updating it. The file **/var/adm/ptmp** may persist if *passwd_file* is being updated and

- The system crashes or
- **yppasswdd** is killed using **SIGKILL** (see *kill*(1) and *signal*(2)).

File **/var/adm/ptmp** must be removed before **yppasswdd** can function properly again.

vipw also uses **/var/adm/ptmp** when updating **/etc/passwd** (see *vipw*(1M)). As a result, **yppasswdd** competes with **vipw** when it updates *passwd_file* if *passwd_file* is **/etc/passwd**.

AUTHOR

yppasswdd was developed by Sun Microsystems, Inc.

FILES

/var/adm/ptmp
lock file used when updating *passwd_file*

SEE ALSO

domainname(1), yppasswd(1), vipw(1M), ypmake(1M), yppasswd(3N), passwd(4), publickey(4), ypfiles(4).

NAME

yppoll - query NIS server for information about NIS map

SYNOPSIS

/usr/sbin/yppoll [**-h** *host*] [**-d** *domain*] *mapname*

Remarks

The Network Information Service (NIS) was formerly known as Yellow Pages (yp). Although the name has changed, the functionality of the service remains the same.

DESCRIPTION

yppoll asks a Network Information Service (NIS) server process (see *ypserv*(1M)) to return the order number (the **time** in seconds when the map was built – *time*(2)) and master NIS server's host name for a NIS database named *mapname*. **yppoll** then writes them to standard output. If the server uses Version 1 NIS protocol, **yppoll** uses this older protocol to communicate with it. **yppoll** also prints the old style diagnostic messages in case of failure.

See *ypfiles*(4) and *ypserv*(1M) for an overview of Network Information Service.

Options

-h *host*	Ask the **ypserv** process on *host* to return the map information (see *ypserv*(1M)). If **-h** *host* is not specified, the host returned by **ypwhich** is used (see *ypwhich*(1)).
-d *domain*	Use *domain* instead of the domain returned by **domainname** (see *domainname*(1)).

AUTHOR

yppoll was developed by Sun Microsystems, Inc.

SEE ALSO

domainname(1), ypwhich(1), ypserv(1M), ypfiles(4).

NAME

yppush - force propagation of Network Information Service database

SYNOPSIS

/usr/sbin/yppush [**-d** *domain*] [**-m** *maxm*] [**-t** *mint*] [**-v**] *mapname*

Remarks

The Network Information Service (NIS) was formerly known as Yellow Pages (yp). Although the name has changed, the functionality of the service remains the same.

DESCRIPTION

yppush copies a Network Information Service (NIS) map (database), *mapname*, from the map's master NIS server to each slave NIS server. It is usually executed only on the master NIS server by shell script **ypmake** which is run either after changes are made to one or more of the master's NIS databases or when the NIS databases are first created. See *ypmake*(1M) and *ypinit*(1M) for more information on these processes.

yppush constructs a list of NIS server host names by reading the NIS map **ypservers** within the *domain*. Keys within the **ypservers** map are the host names of the machines on which the NIS servers run. **yppush** then sends a "transfer map" request to the NIS server at each host, along with the information needed by the transfer agent (the program that actually moves the map) to call back **yppush**.

When the transfer attempt is complete, whether successful or not, and the transfer agent sends **yppush** a status message, the results can be printed to standard output. Messages are printed when a transfer is not possible, such as when the request message is undeliverable or when the timeout period on responses expires.

Refer to *ypfiles*(4) and *ypserv*(1M) for an overview of Network Information Service.

Options

yppush recognizes the following options:

-d *domain* Copy *mapname* to the NIS servers in *domain* rather than to the *domain* returned by **domainname** (see *domainname*(1)).

-m *maxm* Attempt to run *maxm* transfers in parallel to as many servers simultaneously. Without the **-m** option, **yppush** attempts to transfer a map to each server, one at a time. When a network has many servers, such serial transfers can result in long delays to complete all transfers. A *maxm* value greater than 1 reduces total transfer time through better utilization of CPU time at the master. *maxm* can be any value from 1 through the number of NIS servers in the domain.

-t *mint* Set the minimum timeout value to *mint* seconds. When transferring to one slave at a time, **yppush** waits up to 80 seconds for the transfer to complete, after which it begins transferring to the next slave. When multiple parallel transfers are attempted by use of the **-m** option, it may be necessary to set the transfer timeout limit to a value larger than the default 80 seconds to prevent timeouts caused by network delays related to parallel transfers.

-v Verbose mode: messages are printed when each server is called and when each response is received. If this option is omitted, only error messages are printed.

WARNINGS

In the current implementation (Version 2 NIS protocol), the transfer agent is *ypxfr*(1M) which is started by the *ypserv*(1M) program at *yppush*'s request (see *ypxfr*(1M) and *ypserv*(1M)). If *yppush* detects it is interacting with a Version 1 NIS protocol server, it uses the older protocol to send a Version 1 **YPPROC_GET** request and issues a message to that effect. Unfortunately, there is no way of knowing if or when the map transfer is performed for Version 1 servers. **yppush** prints a comment saying that a Version 1 message was sent. The system administrator should then verify by other means that the transfer actually occurred.

AUTHOR

yppush was developed by Sun Microsystems, Inc.

FILES

/usr/sbin/*domain***/ypservers.{dir, pag}**

/usr/sbin/*domain***/***mapname***.{dir, pag}**

SEE ALSO

domainname(1), ypserv(1M), ypxfr(1M), ypfiles(4).

NAME

ypserv, ypbind, ypxfrd - Network Information Service (NIS) server, binder, and transfer processes

SYNOPSIS

/usr/lib/netsvc/yp/ypserv [**-l** *log_file*]

/usr/lib/netsvc/yp/ypbind [**-l** *log_file*] [**-s**] [**-ypset**|**-ypsetme**]

/usr/sbin/ypxfrd

Remarks

The Network Information Service (NIS) was formerly known as Yellow Pages (YP). The functionality remains the same; only the name has changed.

DESCRIPTION

The Network Information Service (NIS) provides a simple network lookup service consisting of databases and processes. The databases are files in a directory tree rooted at **/var/yp** (see *ypfiles*(4)). The processes are **/usr/lib/netsvc/yp/ypserv**, the NIS database lookup server, and **/usr/lib/netsvc/yp/ypbind**, the NIS binder. Both **ypserv** and **ypbind** are daemon processes activated at system startup time when the **NIS_MASTER_SERVER** or **NIS_SLAVE_SERVER** variable is set to 1, for **ypserv,** and the **NIS_CLIENT** variable is set to 1, for **ypbind,** in the **/etc/rc.config.d/namesvrs** file.

The NIS programmatic interface is described in *ypclnt*(3C). Administrative tools are described in *ypwhich*(1), *yppoll*(1M), *yppush*(1M), *ypset*(1M) and *ypxfr*(1M). Tools to see the contents of NIS maps (databases) are described in *ypcat*(1) and *ypmatch*(1). Database generation and maintenance tools are described in *makedbm*(1M), *ypinit*(1M), and *ypmake*(1M). The command to set or show the default NIS domain is *domainname*(1).

ypxfrd transfers entire NIS maps in an efficient manner. For systems that use this daemon, map transfers will be faster, depending on the map. **ypxfrd** should be run on a server running HP-UX release 10.0. **ypxfr** (see *ypxfr*(1M)) will attempt to use **ypxfrd** first. If that fails, it will use the older transfer method. The **ypxfrd** daemon is activated at system startup time when the **NIS_MASTER_SERVER** or **NIS_SLAVE_SERVER** variable is set to 1 in the **/etc/rc.config.d/namesvrs** file.

The **ypserv** daemon's primary function is to look up information in its local collection of NIS maps. It runs only on NIS server machines providing data from NIS databases. Communication to and from **ypserv** is by means of RPC. Lookup functions are described in *ypclnt*(3C).

Four lookup functions perform on a specific map within a NIS domain: **Match**, **Get_first**, **Get_next**, and **Get_all**. The **Match** operation matches a key to a record in the database and returns its associated value. The **Get_first** operation returns the first key-value pair (record) from the map, and **Get_next** enumerates (sequentially retrieves) the remainder of the records. **Get_all** returns all records in the map to the requester as the response to a single RPC request.

Two other functions supply information about the map other than normal map entries: **Get_order_number** and **Get_master_name**. The order number is the time of last modification of a map. The master name is the host name of the machine on which the master map is stored. Both order number and master name exist in the map as special key-value pairs, but the server does not return these through the normal lookup functions. (If you examine the map with **makedbm** or **yppoll** (see *makedbm*(1M) or *yppoll*(1M)), they will be visible.) Other functions are used within the NIS system and are not of general interest to NIS clients. They include:

> **Do_you_serve_this_domain?**
> **Transfer_map**
> **Reinitialize_internal_state**

The **ypbind** daemon remembers information that lets client processes on its machine communicate with a **ypserv** process. The **ypbind** daemon must run on every machine using NIS services, both NIS servers and clients. The **ypserv** daemon may or may not be running on a NIS client machine, but it must be running somewhere on the network or be available through a gateway.

The information that **ypbind** remembers is called a **binding**: the association of a NIS domain name with the Internet address of the NIS server and the port on that host at which the **ypserv** process is listening for service requests. This information is cached in the directory **/var/yp/binding** using a filename in the form *domainname***.***version*.

Client requests drive the binding process. As a request for an unbound domain comes in, the **ypbind** process broadcasts on the network trying to find a **ypserv** process serving maps within that NIS domain. Since the binding is established by broadcasting, at least one **ypserv** process must exist on every network. Once a binding is established for a client, it is given to subsequent client requests. Execute **ypwhich** to query the **ypbind** process (local and remote) for its current binding (see *ypwhich*(1)).

Bindings are verified before they are given to a client process. If **ypbind** is unable to transact with the **ypserv** process it is bound to, it marks the domain as unbound, tells the client process that the domain is unbound, and tries to bind again. Requests received for an unbound domain fail immediately. Generally, a bound domain is marked as unbound when the node running **ypserv** crashes or is overloaded. In such a case, **ypbind** binds to any NIS server (typically one that is less heavily loaded) that is available on the network.

The **ypbind** daemon also accepts requests to set its binding for a particular domain. **ypset** accesses the **Set_domain** facility; it is for unsnarling messes and is not for casual use.

Options

ypserv recognizes the following options:

-l *log_file* Log diagnostic and error messages to the file, *log_file*.

If **ypserv** is started without the **-l** option, **ypserv** writes its messages to **/var/yp/ypserv.log** if that file exists.

If **ypbind** is started without the **-l** option, **ypbind** writes its messages directly to the system console, **/dev/console**.

Information logged to the file includes the date and time of the message, the host name, the process id and name of the function generating the message, and the message itself. Note that different services can share a single log file since enough information is included to uniquely identify each message.

ypbind recognizes the following options:

-l *log_file* Log diagnostic and error messages to the file, *log_file*. See the description above.

-s Secure. When specified, only NIS servers bound to a reserved port are used. This allows for a slight increase in security in completely controlled environments, where there are no computers operated by untrusted individuals. It offers no real increase in security.

-ypset Allow **ypset** to be used to change the binding (see *ypset*(1M)). For maximum security, this option should be used only when debugging the network from a remote machine.

-ypsetme Allow **ypset** to be issued from this machine (see *ypset*(1M)). Security is based on IP address checking, which can be defeated on networks where untrusted individuals may inject packets. This option is not recommended.

AUTHOR

ypserv, **ypbind**, and **ypxfrd** were developed by Sun Microsystems, Inc.

FILES

/var/yp/binding/*domainname***.***version*
These files cache the last successful binding created for the given domain, in order to to speed up the binding process. When a binding is requested, these files are checked for validity and then used.

/var/yp/securenets This file is read by **ypxfrd** and **ypserv**. It contains a list of IP addresses that these servers will allow a binding to.

/var/yp/secureservers This file is read by ypbind. It contains a list of IP addresses that ypbind will receive a binding from.

SEE ALSO

domainname(1), ypcat(1), ypmatch(1), yppasswd(1), ypwhich(1), makedbm(1M), rpcinfo(1M), ypinit(1M), ypmake(1M), yppasswdd(1M), yppoll(1M), yppush(1M), ypset(1M), ypxfr(1M), ypclnt(3C), yppasswd(3N), ypfiles(4).

NAME

ypset - bind to particular Network Information Service server

SYNOPSIS

/usr/sbin/ypset [**-V1**] [**-h** *host*] [**-d** *domain*] *server*

Remarks

The Network Information Service (NIS) was formerly known as Yellow Pages (yp). Although the name has changed, the functionality of the service remains the same.

DESCRIPTION

ypset tells **ypbind** to get Network Information Service (NIS) services for the specified *domain* from the **ypserv** process running on *server* (see *ypserv*(1M) and *ypbind*(1M)). *server* is the NIS server that the NIS client binds to, and is specified as either a host name or an IP address. If *server* is down or is not running **ypserv**, this is not discovered until a local NIS client process tries to obtain a binding for the domain. The **ypbind** daemon then tests the binding set by **ypset**. If the binding cannot be made to the requested server, **ypbind** attempts to rebind to another server in the same domain.

The **ypset** command is useful for binding a client node that is not on a broadcast network, since broadcasting is the method by which **ypbind** locates a NIS server. If a client node exists on a broadcast network which has no NIS server running, and if there is a network with one running that is available via a gateway, **ypset** can establish a binding through that gateway. It is also useful for debugging NIS client applications such as when a NIS map exists only at a single NIS server.

In cases where several hosts on the local net are supplying NIS services, it is possible for **ypbind** to rebind to another host, even while you attempt to find out if the **ypset** operation succeeded. For example, typing **ypset host1** followed by **ypwhich** and receiving the reply **host2** may be confusing. It could occur when *host1* does not respond to **ypbind** because its **ypserv** process is not running or is overloaded, and *host2*, running **ypserv**, gets the binding.

Refer to *ypfiles*(4) and *ypserv*(1M) for an overview of the Network Information Service.

Options

ypset recognizes the following options and command-line arguments:

-V1	Bind *server* for the (old) Version 1 NIS protocol.
-h *host*	Set the binding on *host* instead of locally. *host* can be specified as a host name or an IP address.
-d *domain*	Use *domain* instead of the default domain returned by **domainname** (see *domainname*(1)).

DIAGNOTICS

Sorry, ypbind on host 'name' has rejected your request.

The user is not root, or ypbind was run without the **-ypset** flags. See *ypserv*(1M) for explanations of the **-ypset** flags.

Sorry, I couldn't send my rpc message to ypbind on host 'name'.

The user is not root, or ypbind was run without one of the **-ypset** flags. See *ypserv*(1M) for explanations of the **-ypset** flags.

WARNINGS

The *server* is the NIS server to bind to, specified as either a host name or an IP address. If *server* is a host name, **ypset** uses the NIS services' **hosts** database (built from **/etc/hosts** on the master server) to resolve the name to an IP address. This process works only if the node currently has a valid binding for the domain in question. In most cases, *server* should be specified as an IP address.

AUTHOR

ypset was developed by Sun Microsystems, Inc.

SEE ALSO

domainname(1), ypwhich(1), ypserv(1M), ypfiles(4).

NAME

ypupdated, rpc.ypupdated - server for changing NIS information

SYNOPSIS

/usr/lib/netsvc/yp/rpc.ypupdated [-is]

Remarks

The Network Information Service (NIS) was formerly known as Yellow Pages (yp). Although the name has changed, the functionality of the service remains the same.

DESCRIPTION

ypupdated is a daemon that updates information in the **Network Information Service (NIS)** databases. It is activated at system startup when the **NIS_MASTER_SERVER** variable is set to 1 in **/etc/rc.config.d/namesvrs** file on the NIS master server. **ypupdated** consults the file **updaters** in the directory **/var/yp** to determine which NIS maps should be updated and how to change them.

By default, the daemon requires the most secure method of authentication available to it, either DES (secure) or UNIX (without augmented security).

Options

ypupdated supports the following options:

-i Accept RPC calls with the insecure AUTH_UNIX credentials. This allows programmatic updating of the NIS maps in all networks.

-s Accept only calls authenticated using the secure RPC mechanism (AUTH_DES authentication). This disables programmatic updating of the NIS maps unless the network supports these calls.

AUTHOR

ypupdated was developed by Sun Microsystems, Inc.

FILES

/var/yp/updaters

SEE ALSO

keyserv(1M), updaters(1M).

NAME

ypxfr, ypxfr_1perday, ypxfr_1perhour, ypxfr_2perday - transfer NIS database from server to local node

SYNOPSIS

/usr/sbin/ypxfr [**b**] [**-c**] [**-d** *domain*] [**-f**] [**-h** *host*] [**-s** *domain*] [**-C** *tid prog ipaddr port*] *mapname*

Remarks

The Network Information Service (NIS) was formerly known as Yellow Pages (yp). Although the name has changed, the functionality of the service remains the same.

DESCRIPTION

ypxfr copies a Network Information Service (NIS) map (database) to the local host from a NIS server by using the NIS services. A map can be copied regardless of its age, or it can be copied depending on whether its modification time (order number) is more recent than that of the local map.

The **ypxfr** command creates a temporary map in directory **/var/yp/domain** where *domain* is the NIS *domain*. The **ypxfr** command fills the map with *mapname* entries, obtains the map parameters (master and order number), and loads them. It then clears the old version of *mapname* and moves the temporary map to the existing *mapname*.

If **ypxfr** is run interactively, it writes messages to standard output. If **ypxfr** is invoked without a controlling terminal and if the log file **/var/yp/ypxfr.log** exists, **ypxfr** appends all its messages to that file. Since **ypxfr** is usually run from root's **crontab** file (see *crontab*(1)) or by **yppush** (see *yppush*(1M)), the log file can retain a record of what **ypxfr** attempted and what the results were.

To maintain consistency between NIS servers, **ypxfr** should be executed periodically for every map in the NIS. Different maps change at different rates. For example, the **services.byname** map may not change for months at a time, and might therefore be checked for changes only once a day, such as in the early morning hours. However, **passwd.byname** may change several times per day, so hourly checks for updates might be more appropriate.

A **crontab** file can perform these periodic checks and transfers automatically. Rather than having a separate **crontab** file for each map, **ypxfr** requests can be grouped in a shell script to update several maps at once. Example scripts (mnemonically named) are in **/var/yp**: **ypxfr_1perday**, **ypxfr_2perday**, and **ypxfr_1perhour**. They serve as reasonable rough drafts that can be changed as appropriate.

Refer to *ypfiles*(4) and *ypserv*(1M) for an overview of the Network Information Service.

Options

ypxfr recognizes the following options and command-line arguments:

-b Preserve the resolver flag in the map during transfer.

-c Do not send a "clear current map" request to the local **ypserv** process. Use this flag if **ypserv** is not running locally when you are running **ypxfr**. Otherwise, **ypxfr** complains that it cannot talk to the local **ypserv**, and the transfer fails. If **ypserv** is running locally, do not use this flag.

-d *domain* Copy the map from a NIS server in *domain* rather than the *domain* returned by **domainname** (see *domainname*(1)).

-f Force the map to be copied, even if its order number at the remote NIS server is not more recent than the order number of the local map.

-h *host* Obtain the map from *host*, regardless of its master server. If this option is not used, **ypxfr** asks the NIS service for the master's host name and tries to obtain its map. The *host* can be a name or an IP address of the form *a.b.c.d*.

-s *domain* Specify a source domain from which to transfer a map that should be the same across domains (such as the **services.byname** map.

-C *tid prog ipaddr port*
This option is used only by **ypserv**. When **ypserv** invokes **ypxfr**, it specifies that **ypxfr** should call back a **yppush** process (that initiated the transfer) at the host with IP address *ipaddr*, registered as program number *prog*, listening on port *port*, and waiting for a response to transaction *tid*.

AUTHOR

ypxfr was developed by Sun Microsystems, Inc.

FILES

/usr/sbin/ypxfr.log log file

The following scripts are suggested for use with **cron**.

/usr/sbin/ypxfr_1perday
run one transfer per day

/usr/sbin/ypxfr_2perday
run two transfers per day

/usr/sbin/ypxfr_1perhour
hourly transfers of "volatile" maps

SEE ALSO

crontab(1), domainname(1), cron(1M), ypinit(1M), yppush(1M), ypserv(1M), ypfiles(4).