Files Reference
Files Reference
Note
Before using this information and the product it supports, read the information in "Notices," on page 969.

Fifth Edition (November 2007)

This edition applies to AIX 5L Version 5.3 and to all subsequent releases of this product until otherwise indicated in new editions.

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

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

### About This Book
- How to Use This Book
- Highlighting
- Case-Sensitivity in AIX
- ISO 9000
- Related Publications

### Chapter 1. System Files
- Types of Files
- File-Naming Conventions
- System Files
- Related Information
- acct.cfg File
- admin File
- aliases File for Mail
- audit File for BNU
- backup File
- bincmds File
- BOOTP Relay Agent Configuration File
- bootparams File for NFS
- ca.cfg File
- cdromd.conf File Format
- ClientHostName.info File
- clsnmp.conf File
- Command (C.*) Files for BNU
- compver File
- config File
- consdef File
- copyright File
- ct_has.pkt File
- ct_has.qkf File
- ct_has.thl File
- ctcasd.cfg File
- ctrmc.acls File
- ctsec_map.global File
- ctsec_map.local File
- Data (D.*) Files for BNU
- /dev/hty File
- /dev/rhp File
- DHCP Client Configuration File
- DHCP Server Configuration File
- depend File
- dir File
- dpid2.conf File
- dsinfo File
- dumpdates File
- e789_ctbl File for HCON
- e789_ktbl File for HCON
- eimadmin.conf File
- environ File
- environment File
- errors File for BNU
- ethers File for NIS

---

© Copyright IBM Corp. 1997, 2007
<table>
<thead>
<tr>
<th>File Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>events File</td>
<td>75</td>
</tr>
<tr>
<td>Execute (X.*) Files for BNU</td>
<td>77</td>
</tr>
<tr>
<td>exports File for NFS</td>
<td>79</td>
</tr>
<tr>
<td>.fig File</td>
<td>85</td>
</tr>
<tr>
<td>filesystems File</td>
<td>85</td>
</tr>
<tr>
<td>Foreign File for BNU</td>
<td>87</td>
</tr>
<tr>
<td>.forward File</td>
<td>88</td>
</tr>
<tr>
<td>ftpaccess.ctl File</td>
<td>89</td>
</tr>
<tr>
<td>/etc/group File</td>
<td>90</td>
</tr>
<tr>
<td>/etc/security/group File</td>
<td>91</td>
</tr>
<tr>
<td>Workload Manager groupings File</td>
<td>93</td>
</tr>
<tr>
<td>hostmibd.conf File</td>
<td>94</td>
</tr>
<tr>
<td>image.data File</td>
<td>95</td>
</tr>
<tr>
<td>/etc/security/.ids File</td>
<td>99</td>
</tr>
<tr>
<td>Ined Files</td>
<td>100</td>
</tr>
<tr>
<td>.info File</td>
<td>101</td>
</tr>
<tr>
<td>initab File</td>
<td>103</td>
</tr>
<tr>
<td>irs.conf File</td>
<td>105</td>
</tr>
<tr>
<td>ispaths File</td>
<td>109</td>
</tr>
<tr>
<td>isprime File</td>
<td>112</td>
</tr>
<tr>
<td>.kshrc File</td>
<td>112</td>
</tr>
<tr>
<td>ldap_subroutines Information</td>
<td>114</td>
</tr>
<tr>
<td>ldapid.ldif.template File</td>
<td>116</td>
</tr>
<tr>
<td>limits File</td>
<td>117</td>
</tr>
<tr>
<td>local_domain File</td>
<td>119</td>
</tr>
<tr>
<td>login.cfg File</td>
<td>120</td>
</tr>
<tr>
<td>lpacl Information</td>
<td>123</td>
</tr>
<tr>
<td>.maildelivery File for MH</td>
<td>131</td>
</tr>
<tr>
<td>/usr/lib/security/methods.cfg File</td>
<td>134</td>
</tr>
<tr>
<td>mhl.format File</td>
<td>136</td>
</tr>
<tr>
<td>.mh_profile File</td>
<td>139</td>
</tr>
<tr>
<td>mibll.my File</td>
<td>143</td>
</tr>
<tr>
<td>mkuser.default File</td>
<td>145</td>
</tr>
<tr>
<td>mtstailor File for MH</td>
<td>146</td>
</tr>
<tr>
<td>mrouted.conf File</td>
<td>147</td>
</tr>
<tr>
<td>netgroup File for NIS</td>
<td>149</td>
</tr>
<tr>
<td>netmasks File for NIS</td>
<td>150</td>
</tr>
<tr>
<td>netsvc.conf File</td>
<td>151</td>
</tr>
<tr>
<td>networks File for NFS</td>
<td>153</td>
</tr>
<tr>
<td>NLSvec File</td>
<td>153</td>
</tr>
<tr>
<td>ntp.conf File</td>
<td>155</td>
</tr>
<tr>
<td>ntp.keys File</td>
<td>164</td>
</tr>
<tr>
<td>objects File</td>
<td>165</td>
</tr>
<tr>
<td>pam_aix Module</td>
<td>166</td>
</tr>
<tr>
<td>pam_allow Module</td>
<td>168</td>
</tr>
<tr>
<td>pam_allowroot Module</td>
<td>169</td>
</tr>
<tr>
<td>pam_ckfile Module</td>
<td>170</td>
</tr>
<tr>
<td>pam_permission Module</td>
<td>171</td>
</tr>
<tr>
<td>pam_prohibit Module</td>
<td>172</td>
</tr>
<tr>
<td>pam_rhosts_auth Module</td>
<td>173</td>
</tr>
<tr>
<td>pam.conf File</td>
<td>174</td>
</tr>
<tr>
<td>/etc/passwd File</td>
<td>176</td>
</tr>
<tr>
<td>passwd_policy File</td>
<td>178</td>
</tr>
<tr>
<td>/etc/security/passwd File</td>
<td>179</td>
</tr>
<tr>
<td>pcnfsd.conf Configuration File</td>
<td>181</td>
</tr>
<tr>
<td>pkginfo File</td>
<td>184</td>
</tr>
</tbody>
</table>
Chapter 2. File Formats

Asynchronous Terminal Emulation (ATE) File Formats

Basic Networking Utilities (BNU) File Formats

tip File Formats

TCP/IP System Management File Formats
profile File Format ................................................. 546
protocols File Format for TCP/IP .......................... 547
queuedefs File Format ........................................ 548
rc.net File Format for TCP/IP .............................. 549
rc.ntx File Format .............................................. 552
remote File Format for tip ................................ 554
resolv.conf File Format for TCP/IP ...................... 557
resolv.ldap File Format for TCP/IP ...................... 559
.rhosts File Format for TCP/IP ............................. 560
sccsfile File Format ........................................... 562
services File Format for TCP/IP ............................ 566
setmaps File Format ............................................ 567
simprof File Format ............................................ 568
Standard Resource Record Format for TCP/IP .... 570
Sysfiles File Format for BNU ............................... 578
Systems File Format for BNU ............................... 579
telnet.conf File Format for TCP/IP ..................... 586
terminfo Directory .............................................. 587
.tipc File Format for tip ................................... 641
trcfmt File Format ............................................. 642
troff File Format .............................................. 647
troff Font File Format ....................................... 649
tunables File Format ......................................... 652
uconvdef Source File Format ............................. 654
UIL File Format ................................................ 656
utmp, wtmp, failedlogin File Format .................. 677
vgrindefs File Format ......................................... 677
XCOFF Object File Format ................................. 685

Chapter 3. Special Files ........................................ 751
Related Information ............................................. 752
3270cn Special File ............................................ 752
bus Special File ............................................... 758
cd Special File ................................................ 759
console Special File .......................................... 760
dials Special File ............................................. 763
dump Special File ............................................. 764
entn Special File .............................................. 765
Error Logging Special Files ................................ 766
fd Special File .................................................. 767
fddin Special File .............................................. 769
GIO Special File ............................................... 771
ide Special File ............................................... 772
kbd Special File ............................................... 772
lft Special File ................................................ 774
lp Special File .................................................. 775
lpfk Special File .............................................. 778
lvdd Special File .............................................. 779
mem or kmem Special File ................................. 783
mouse Special File ............................................ 786
mpcn Special File ............................................. 787
mpqi Special File ............................................. 789
mpqn Special File ............................................. 790
null Special File ................................................ 791
nvram Special File ............................................ 792
random and urandom Devices .................................................. 793
omd Special File .................................................................. 794
opn Special File .................................................................. 797
ops0 Special File .................................................................. 797
pty Special File .................................................................. 799
rcm Special File .................................................................. 801
rhdisk Special File ............................................................... 801
rmt Special File .................................................................. 803
scsi Special File .................................................................. 807
tablet Special File ............................................................... 807
tmscsi Special File ............................................................... 809
tokn Special File .................................................................. 809
trace Special File ................................................................ 811
tty Special File .................................................................. 812
urandom and random Devices ............................................... 812
vty_server Special File ......................................................... 813
x25sn Special File ................................................................ 815

Chapter 4. Header Files .......................................................... 819

Related Information ................................................................ 820
List of Major Control Block Header Files ................................. 820
cf_ffdc.h File ..................................................................... 822
dirent.h File ....................................................................... 824
difcn.h File ......................................................................... 825
eucioctl.h File ..................................................................... 825
cfcntl.h File ...................................................................... 826
filsys.h File ....................................................................... 828
flock.h File ......................................................................... 831
fullstat.h File ..................................................................... 833
grp.h File ........................................................................... 834
iconv.h File ........................................................................ 835
inode.h File ........................................................................ 835
inttypes.h File ..................................................................... 839
ipc.h File ............................................................................ 840
iso646.h File ..................................................................... 840
ldr.h File ............................................................................ 841
limits.h File ........................................................................ 843
libperfstat.h File ................................................................ 845
math.h File ......................................................................... 859
mode.h File ......................................................................... 860
msg.h File ........................................................................... 862
mtio.h File .......................................................................... 864
param.h File ....................................................................... 864
pmapi.h File ....................................................................... 865
poll.h File ........................................................................... 871
pthread.h File ..................................................................... 872
pwd.h File ........................................................................... 875
pwdpolicy.h File ................................................................. 875
sem.h File ............................................................................ 877
gtgty.h File ....................................................................... 879
shm.h File .......................................................................... 885
spc.h File ............................................................................ 886
srcobj.h File ...................................................................... 890
stat.h File ........................................................................... 891
statfs.h File ....................................................................... 893
statvfs.h File ...................................................................... 895
About This Book

This book provides experienced programmers with complete, detailed information about files for the AIX operating system. Files are listed alphabetically, and complete descriptions are given for each file. It includes information on file formats, special files, system files, header files, directories, and related information associated with files. This publication is also available on the documentation CD that is shipped with the operating system.

How to Use This Book

This book contains sections on the system files, special files, header files, and directories that are provided with the operating system and optional program products. File formats required for certain files that are generated by the system or an optional program are also presented in a section of this book.

Highlighting

The following highlighting conventions are used in this book:

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

Case-Sensitivity in AIX

Everything in the AIX operating system is case-sensitive, which means that it distinguishes between uppercase and lowercase letters. For example, you can use the `ls` command to list files. If you type `LS`, the system responds that the command is “not found.” Likewise, `FILEA`, `FiLea`, and `filea` are three distinct file names, even if they reside in the same directory. To avoid causing undesirable actions to be performed, always ensure that you use the correct case.

ISO 9000

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

Related Publications

The following books contain information about or related to the operating system files:

- *Operating system and device management*
- *Networks and communication management*
- *AIX 5L Version 5.3 Commands Reference* (six volumes)
- *AIX 5L Version 5.3 Technical Reference* (six volumes)
- *AIX 5L Version 5.3 General Programming Concepts: Writing and Debugging Programs*
- *AIX 5L Version 5.3 Communications Programming Concepts*
- *GL3.2 Version 4.1 for AIX: Programming Concepts*
Chapter 1. System Files

A file is a collection of data that can be read from or written to. A file can be a program you create, text you write, data you acquire, or a device you use. Commands, printers, terminals, and application programs are all stored in files. This allows users to access diverse elements of the system in a uniform way and gives the operating system great flexibility. No format is implied when a file is created.

Files are used for all input and output (I/O) of information in this operating system. This standardizes access to both software and hardware. Input occurs when the content of a file is modified or written to. Output occurs when the content of one file is read or transferred to another file. For example, to create a hardcopy printout of a text file, the system reads the information from the text file and writes the data to the file representing the printer.

Collections of files are stored in directories. These collections of files are often related to each other, and storing them in a structure of directories keeps them organized.

There are many ways to create, use, and manipulate files. Files in Operating system and device management introduces the commands that control files.

Types of Files

There are three basic types of files:

<table>
<thead>
<tr>
<th>File Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>regular</td>
<td>Stores data (text, binary, and executable).</td>
</tr>
<tr>
<td>directory</td>
<td>Contains information used to access other files.</td>
</tr>
<tr>
<td>special</td>
<td>Defines a FIFO (first-in, first-out) file or a physical device.</td>
</tr>
</tbody>
</table>

All file types recognized by the system fall into one of these categories. However, the operating system uses many variations of these basic types.

Regular files are the most common. When a word processing program is used to create a document, both the program and the document are contained in regular files.

Regular files contain either text or binary information. Text files are readable by the user. Binary files are readable by the computer. Binary files can be executable files that instruct the system to accomplish a job. Commands, shell scripts, and other programs are stored in executable files.

Directories contain information the system needs to access all types of files, but they do not contain the actual file data. As a result, directories occupy less space than a regular file and give the file-system structure flexibility and depth. Each directory entry represents either a file or subdirectory and contains the name of a file and the file’s i-node (index node reference) number. The i-node number represents the unique i-node that describes the location of the data associated with the file. Directories are created and controlled by a separate set of commands. See Directories in Operating system and device management for more information.

Special files define devices for the system or temporary files created by processes. There are three basic types of special files: FIFO (first-in, first-out), block, and character. FIFO files are also called pipes. Pipes are created by one process to temporarily allow communication with another process. These files cease to exist when the first process finishes. Block and character files define devices.

Every file has a set of permissions (called access modes) that determine who can read, modify, or execute the file. To learn more about file access modes, see File ownership and user groups in Operating system and device management.
File-Naming Conventions

The name of each file must be unique within the directory where it is stored. This insures that the file also has a unique path name in the file system. File-naming guidelines are:

- A file name can be up to 255 characters long and can contain letters, numbers, and underscores.
- The operating system is case-sensitive which means it distinguishes between uppercase and lowercase letters in file names. Therefore, FILEA, FiLea, and filea are three distinct file names, even if they reside in the same directory.
- File names should be as descriptive as possible.
- Directories follow the same naming conventions as files.
- Certain characters have special meaning to the operating system, and should be avoided when naming files. These characters include the following:
  / \ " ' * ; - ? [ ] ( ) " " ! $ { } < > @ & |
- A file name is hidden from a normal directory listing if it begins with a . (dot). When the ls command is entered with the -a flag, the hidden files are listed along with regular files and directories.

The path name of a file consists of the name of every directory that precedes it in the file tree structure. Only the final component of a path name can contain the name of a regular file. All other components in a path name must be directories. Path names can be absolute or relative. See File path names in Operating system and device management to learn more about the complete name of a file within the file system.

System Files

The files in the following chapter are system files. These files are created and maintained by the operating system and are necessary for the system to perform its many functions. System files are used by many commands and subroutines to perform operations. These files can only be changed by a user with root authority.

Related Information

Files in Operating system and device management introduces the basic concepts of files and directories and the commands that control them.

Files, Directories, and File Systems for Programmers in AIX 5L Version 5.3 General Programming Concepts: Writing and Debugging Programs introduces i-nodes, file space allocation, and the file, directory, and file system subroutines.

acct.cfg File

Purpose

The acct.cfg file consists of CA stanzas and LDAP stanzas. The CA stanzas contain private CA information not suitable for the publicly readable ca.cfg file. LDAP stanzas contain LDAP information such as LDAP administrative names and passwords.

Description

For every CA stanza in the ca.cfg file, the acct.cfg file should contain an equivalently named CA stanza, and all CA stanzas must be uniquely named. On the other hand, all LDAP stanzas are named ldap. For this reason, a CA stanza cannot be named ldap. Also, no stanza can be named default. An LDAP stanza must exist and at least one CA stanza, named local must exist.
Examples

CA Stanzas:

- **carefnum**: Specifies the CA's reference number used while communicating with the CA through CMP. This value must be the same value as the one that is specified while configuring the CA. (Required)
- **capasswd**: Specifies the CA's password used while communicating with the CA. The length of the password must be at least 12 characters long. This value must be the same value as the one that is specified while configuring the CA. (Required)
- **rvrefnum**: Specifies the revocation reference number used for revoking a certificate
- **rvpasswd**: Specifies the revocation password used for CMP. The length of the password must be at least 12 character long.
- **keylabel**: Defines the name of the key label in the trusted keystore. (Required)
- **keypasswd**: Defines the password of the trusted keystore. (Required)

ldap Stanzas:

- **ldappkiadmin**: Specifies the PKI LDAP administrator account name.
- **ldappkiadmpwd**: Specifies the PKI LDAP administrator account password.
- **ldapservers**: Specifies the LDAP server machine name or IP address.
- **ldapsuffix**: Specifies the LDAP DN suffix for the root of the LDAP branch where the PKI data resides.

Local:

- carefnum = 12345678
- capasswd = password1234
- rvrefnum = 9999997
- rvpasswd = password
- keylabel = "Trusted Key"
- keypasswd = somepassword

Ldap:

- ldappkiadmin = "cn=admin"
- ldappkiadmpwd = password
- ldapservers = myserver.mydomain.com
- ldapsuffix = "ou=cert,cn=aixsecdb"

File

/usr/lib/security/pki/acct.cfg

Related Information

The **certcreate** command.

The **certrevoke** command.

The **/usr/lib/security/pki/ca.cfg** file.
admin File

Purpose
Describes the format of an installation defaults file.

Description
admin is a generic name for an ASCII file that defines default installation actions by assigning values to installation parameters. For example, it allows administrators to define how to proceed when the package being installed already exists on the system.

/var/sadm/install/admin/default is the default admin file delivered with your system. The default file is not writable, so to assign values different from this file, create a new admin file. There are no naming restrictions for admin files. Name the file when installing a package with the -a flag of the pkgadd command. If the -a flag is not used, the default admin file is used.

Each entry in the admin file is a line that establishes the value of a parameter in the following form:

    param=value

Eleven parameters can be defined in an admin file. A file is not required to assign values to all eleven parameters. If a value is not assigned, pkgadd asks the installer how to proceed.

The eleven parameters and their possible values are shown below except as noted. They may be specified in any order. Any of these parameters can be assigned the value ask, which means that, if the situation occurs, the installer is notified and asked to supply instructions at that time.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>basedir</td>
<td>Indicates the base directory where relocatable packages are to be installed. The value may contain $PKGINST to indicate a base directory that is to be a function of the package instance.</td>
</tr>
<tr>
<td>mail</td>
<td>Defines a list of users to whom mail should be sent following installation of a package. If the list is empty or if the parameter is not present in the admin file, the default value of root is used. The ask value cannot be used with this parameter.</td>
</tr>
<tr>
<td>runlevel</td>
<td>Indicates resolution if the run level (system state) is not correct for the installation or removal of a package. Options are:</td>
</tr>
<tr>
<td></td>
<td>nocheck: Do not check for run level (system state).</td>
</tr>
<tr>
<td></td>
<td>quit: Abort installation if run level (system state) is not met.</td>
</tr>
<tr>
<td>conflict</td>
<td>Specifies what to do if an installation expects to overwrite a previously installed file, thus creating a conflict between packages. Options are:</td>
</tr>
<tr>
<td></td>
<td>nocheck: Do not check for conflict; files in conflict are overwritten.</td>
</tr>
<tr>
<td></td>
<td>quit: Abort installation if conflict is detected.</td>
</tr>
<tr>
<td></td>
<td>nochange: Override installation of conflicting files; conflicting files are not installed.</td>
</tr>
<tr>
<td>setuid</td>
<td>Checks for executables that have setuid or setgid bits enabled after installation. Options are:</td>
</tr>
<tr>
<td></td>
<td>nocheck: Do not check for setuid executables.</td>
</tr>
<tr>
<td></td>
<td>quit: Abort installation if setuid processes are detected.</td>
</tr>
<tr>
<td></td>
<td>nochange: Override installation of setuid processes; processes are installed without setuid bits enabled.</td>
</tr>
</tbody>
</table>
Parameter  | Description
---------- | ------------
action     | Determines if action scripts provided by package developers contain possible security impact. Options are:

nocheck    | Ignore security impact of action scripts.
quit       | Abort installation if action scripts may have a negative security impact.

partial    | Checks to see if a version of the package is already partially installed on the system. Options are:

nocheck    | Do not check for a partially installed package.
quit       | Abort installation if a partially installed package exists.

idepend    | Controls resolution if other packages depend on the one to be installed. Options are:

nocheck    | Do not check package dependencies.
quit       | Abort installation if package dependencies are not met.

rdepend    | Controls resolution if other packages depend on the one to be removed. Options are:

nocheck    | Do not check package dependencies.
quit       | Abort removal if package dependencies are not met.

space      | Controls resolution if disk space requirements for package are not met. Options are:

nocheck    | Do not check space requirements (installation fails if it runs out of space).
quit       | Abort installation if space requirements are not met.

The value ask cannot be defined in an admin file that is used for non-interactive installation (since by definition, there is no installer interaction). Doing so causes installation to fail when input is needed.

Related Information

The pkgadd command.

aliases File for Mail

Purpose
Contains alias definitions for the sendmail command.

Description
The /etc/mail/aliases file contains the required aliases for the sendmail command. Do not change these defaults, as they are required by the system. The file is formatted as a series of lines in the form:

name: name_1, name_2, name_3,...

The name: is the name of the alias, and the name_n are the aliases for that name. Lines beginning with white space are continuation lines. Lines beginning with a # (pound sign) are comments.

Aliasing occurs only on local names. System-wide aliases are used to redirect mail. For example, if you receive mail at three different systems, you can use the /etc/mail/aliases file to redirect your mail to one of the systems. As an individual user, you can also specify aliases in your .mailrc file.

Aliases can be defined to send mail to a distribution list. For example, you can send mail to all of the members of a project by sending mail to a single name.
The sender of a message is not included when the sendmail command expands an alias address. For example, if amy sends a message to alias D998 and she is defined as a member of that alias, the sendmail command does not send a copy of the message to amy.

The /etc/mail/aliases file is a raw data file. The sendmail command uses a database version of this file. You must build a new alias database by running the sendmail -bi command or the newaliases command before any changes made to the /etc/mail/aliases file become effective.

Berkeley DB support is now available on AIX 5.1 for Sendmail 8.11.0. As long as you do not rebuild the aliases database, sendmail will continue to read it in its old DBM format. This consists of two files: /etc/mail/aliases.dir and /etc/mail/aliases.pag. However, the moment you rebuild the aliases database, sendmail will change this format to Berkeley DB. This file will be stored in /etc/mail/aliases.db.

Note: Upper case characters on the left hand side of the alias are converted to lowercase before being stored in the aliases database. In the following example, mail sent to the testalias user alias fails, since TEST is converted to test when the second line is stored.

TEST: user@machine
testalias: TEST

To preserve uppercase in user names and alias names, add the u flag to the local mailer description in the /etc/mail/sendmail.cf file. Thus, in the example above, mail to the testalias user alias would succeed.

Files

/etc/mail/aliases Contains systemwide aliases.
/etc/mail/aliasesDB directory Contains the binary files created by the newaliases command, including the DB.dir and DB.pag files.
/etc/mail/aliases.db Contains the binary file storing the aliases database in Berkeley DB format, created by the newaliases command

Related Information

The newaliases command, sendmail command.
The .mailrc file.

audit File for BNU

Purpose
Contains debug messages from the uucico daemon.

Description
The /var/spool/uucp/.Admin/audit file contains debug messages from the uucico daemon when it is invoked as a result of a call from another system. If the uucico daemon is invoked from the local system, the debug messages are sent to either the /var/spool/uucp/.Admin/errors file or to standard output.

Files

/var/spool/uucp/.Admin/audit Specifies the path of the audit file.
/var/spool/uucp/.Admin/errors Contains a record of uucico daemon errors.
Related Information

The `uudemon.cleanu` command.

The `cron` daemon, `uucico` daemon.

BNU log files, BNU Daemons, BNU File and Directory Structure in Networks and communication management.

backup File

Purpose
Copies the file system onto temporary storage media.

Description
A backup of the file system provides protection against substantial data loss due to accidents or error. The `backup` command writes file system backups in the `backup` file format, and conversely, the `restore` command reads file system backups. The backup file contains several different types of header records along with the data in each file that is backed up.

Header Records
The different types of header records for by-name backups are:

<table>
<thead>
<tr>
<th>Header Record</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FS_VOLUME</td>
<td>Exists on every volume and holds the volume label.</td>
</tr>
<tr>
<td>FS_NAME_X</td>
<td>Holds a description of a file backed up by name.</td>
</tr>
<tr>
<td>FS_END</td>
<td>Indicates the end of the backup. This header appears at the end of the last volume.</td>
</tr>
</tbody>
</table>

The different types of header records for by_inode and name backups are:

<table>
<thead>
<tr>
<th>Header Record</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TS_TAPE</td>
<td>Exists on every volume and holds the volume label.</td>
</tr>
<tr>
<td>TS_BITS</td>
<td>Describes the directory structure.</td>
</tr>
<tr>
<td>TS_CLRI</td>
<td>Describes the unused i-node numbers on the backup system.</td>
</tr>
<tr>
<td>TS_INODE</td>
<td>Describes the file.</td>
</tr>
<tr>
<td>TS_ADDR</td>
<td>Indicates a continuation of the preceding file.</td>
</tr>
<tr>
<td>TS_END</td>
<td>Indicates the end of the backup.</td>
</tr>
</tbody>
</table>

The descriptions of the fields of the header structure for by_inode backups are:

<table>
<thead>
<tr>
<th>Header Record</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>c_type</td>
<td>The header type.</td>
</tr>
<tr>
<td>c_date</td>
<td>The current dump date.</td>
</tr>
<tr>
<td>c_ddate</td>
<td>The file system dump date.</td>
</tr>
<tr>
<td>c_volume</td>
<td>The volume number.</td>
</tr>
<tr>
<td>c_tapea</td>
<td>The number of the current header record.</td>
</tr>
<tr>
<td>c_inumber</td>
<td>The i-node number on this record.</td>
</tr>
<tr>
<td>c_magic</td>
<td>The magic number.</td>
</tr>
<tr>
<td>c_checksum</td>
<td>The value that would make the record sum to the CHECKSUM value.</td>
</tr>
<tr>
<td>bsd_c_dinode</td>
<td>A copy of the BSD i-node as it appears on the BSD file system.</td>
</tr>
<tr>
<td>c_count</td>
<td>The number of characters in the c_addr field.</td>
</tr>
<tr>
<td>c_addr</td>
<td>A character array that describes the blocks being dumped for the file.</td>
</tr>
</tbody>
</table>
Each volume except the last ends with a tape mark (read as an end of file). The last volume ends with a `TS_END` record and then the tape mark.

**By-Name Format**
The format of a by-name backup is:

```
FS_VOLUME
FS_NAME_X (before each file)
File Data
FS_END
```

**By-Inode Format**
The format of a by-inode backup follows:

```
TS_VOLUME
TS_BITS
TS_CLRI
TS_INODE
TS_END
```

A detailed description of the by-inode header file follows:

```c
union u_spcl {
   char dummy[TP_BSIZE];
   struct s_spcl {
      int   c_type;      /*  4 */
      time_t c_date;     /*  8 */
      time_t c_ddate;    /* 12 */
      int   c_volume;    /* 16 */
      daddr_t c_tapea;   /* 20 */
      ino_t  c_inumber;  /* 24 */
      int   c_magic;     /* 28 */
      int   c_checksum;  /* 32 */
      struct bsd_dinode bsd_c_dinode; /* 160 */
      int   c_count;     /* 164 */
      char  c_addr[TP_NINDIR];  /* 676 */
      int   xix_flag;     /* 680 */
      struct dinode xix_dinode; /* 800 */
   } s_spcl;
} u_spcl;
```

**Constants**
Constants used to distinguish these different types of headers and define other variables are:

```c
#define OSF_MAGIC  (int)60011
#define NFS_MAGIC  (int)60012    /* New File System Magic */
#define XIX_MAGIC  (int)60013    /* Magic number for v3 */
```
Related Information

The `backup` command, `pack` command, `restore` command.

The `filesystems` file.

`File systems` and `Backup methods` in *Operating system and device management.*

## bincmds File

### Purpose

Contains the shell commands that process audit bin data.

### Description

The `/etc/security/audit/bincmds` file is an ASCII template file that contains the backend commands that process audit binfile records. The path name of this file is defined in the `bin` stanza of the `/etc/security/audit/config` file.

This file contains command lines each composed of one or more commands with input and output that can be piped together or redirected. Although the commands usually are one or more of the audit system commands (the `auditcat` command, the `auditpr` command, the `auditselect` command), this is not a requirement.

As each bin file is filled by the kernel, the `auditbin` daemon invokes each command to process the bin records, substituting the names of the current bin file and the audit trail file for any `$trail` and `$bin` strings in the commands. Upon startup, if the `auditbin` daemon detects that the bin files require a recovery procedure, the command will prepend a `-r` to the bin file’s name in `$bin`.

**Note:** The commands are executed by the trusted shell (TSH) when on the trusted path. This means that the path names in the commands must be absolute, and that environment variable substitution may be limited. See the discussion of the `tsh` command for more information.

### Security

Access Control: This file should grant read (r) access to the root user and members of the audit group and grant write (w) access only to the root user.

### Examples

1. To compress audit bin records and append them to the system audit trail file, include the following line in the `/etc/security/audit/bincmds` file:

   `/usr/sbin/auditcat -p -o $trail $bin`
When the command runs, the names of the current bin file and the system audit-trail file are substituted for the $bin and $trail strings. Records are compressed and appended to the /audit/trail file.

2. To select the audit events from each bin file that are unsuccessful because of authentication or privilege reasons and append the events to the /audit/trail.violations file, you must include the following line in the /etc/security/audit/bincmds file:

```
/usr/sbin/auditselect -e "result == FAIL_AUTH || result == FAIL_PRIV" $bin >> /audit/trail.violations
```

3. To create a hard-copy audit log of all local user authentication audit events, include the following line in the /etc/security/audit/bincmds file:

```
/usr/sbin/auditselect -e "event == USER_Login || event == USER_SU" $bin | \
/usr/sbin/auditpr -t2 -v /dev/lpr3
```

Adjust the printer name to fit your requirements.

### Files

- /etc/security/audit/bincmds: Specifies the path to the file.
- /etc/security/audit/config: Contains audit-system configuration information.
- /etc/security/audit/events: Contains the audit events of the system.
- /etc/security/audit/objects: Contains audit events for audited objects (files).
- /etc/security/audit/streamcmds: Contains auditstream commands.

### Related Information

The audit, auditcat, auditpr, auditselect, and tsh commands.

The auditbin daemon.

Setting Up Auditing in [Operating system and device management](#).

Security Administration Auditing overview in [Operating system and device management](#).

### BOOTP Relay Agent Configuration File

#### Purpose

Default configuration information for the BOOTP (boot protocol) relay agent program (dhcprd).

#### Description

The dhcpd configuration file contains entries for logging information and servers to receive BOOTP packets.

This file is part of TCP/IP in Network Support Facilities in Base Operating System (BOS) Runtime.

Following are the formats for the data in the configuration file.

**Format**

<table>
<thead>
<tr>
<th>Format</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td># Comment line</td>
<td>The # character means that there is a comment from that point to the end of the line.</td>
</tr>
<tr>
<td>numLogFiles n</td>
<td>Specifies the number of log files. If 0 is specified, no log file will be maintained, and no log message is displayed anywhere. n is the maximum number of log files maintained as the size of the most recent log file reaches its maximum size and a new log file is created.</td>
</tr>
</tbody>
</table>
Format

logFileSize n

Meaning

Maximum size of a log file. When the size of the most recent log file reaches this value, it is renamed and a new log file is created. \( n \) is measured in kilobytes (KB).

logFileName filename

Name and path of the most recent log file. Less recent log files have the number 1 to \((n - 1)\) appended to their names; the larger the number, the older the file.

logItem <option name>

One item that will be logged. Multiple of these lines are allowed. This allows for the specified logging level to be turned on. The following are option names:

SYSERR
  System error, at the interface to the platform

OBJERR
  Object error, in between objects in the process

PROTERR
  Protocol error, between client and server

WARNING
  Warning, worth attention from the user

EVENT
  Event occurred to the process

ACTION
  Action taken by the process

INFO
  Information that might be useful

ACNTING
  Who was served, and when

TRACE
  Code flow, for debugging.

server <ip address>

The address of a server to receive the DHCP or BOOTP packet. Multiple servers may be specified, and all will receive the packet.

Example

The following example sets the logging parameters and configures two servers to receive BOOTP and DHCP packets. The servers are specified singly and with their ip addresses. The logging statements below tell the daemon to use at most four logfiles, rotate the log files after their size is 100 kilobytes of data, and place the files in the local directory and use `dhcpsd.log` as the base name. On rotation, the old file will be moved to `dhcpsd.log1`, and the daemon will start logging to an empty `dhcpsd.log`.

```
numLogFiles  4
logFileSize  100
logFileName  dhcpsd.log
logItem      SYSERR
logItem      OBJERR
logItem      PROTERR
logItem      WARNING
logItem      EVENT
logItem      ACTION
logItem      INFO
logItem      ACNTING
logItem      TRACE

server 129.35.128.43
server 9.3.145.5
```

Related Information

The `dhcpd` Daemon, the `bootpd` Daemon
TCP/IP address and parameter Assignment - Dynamic Host Configuration Protocol in Networks and communication management.

TCP/IP problems with Dynamic Host Configuration Protocol in Networks and communication management.

---

### bootparams File for NFS

**Purpose**
Contains the list of client entries that diskless clients use for booting.

**Description**
The /etc/bootparams file for Network File System (NFS) contains a list of client entries that diskless clients use for booting. The first item of each entry is the name of the diskless client. Each entry should contain the following information:
- Name of client
- List of keys, names of servers, and path names

Items are separated by tab characters.

**Examples**
The following is an example of a /etc/bootparams file:
```
myclient root=myserver:/nfsroot/myclient 
    swap=myserver:/nfsswar/myclient 
    dump=myserver:/nfsdump/myclient
```

**Files**

/etc/bootparams  Specifies the path of the bootparams file.

**Related Information**
Network File System Overview in Networks and communication management.

---

### ca.cfg File

**Purpose**
The ca.cfg file consists of CA stanzas. The CA stanzas contain public CA information used by the Certificate Authentication Services for generating certificate requests and certificate revocation requests.

**Description**
For every CA stanza in the ca.cfg file, the acct.cfg file should contain an equivalently named CA stanza. Each CA stanza name in the ca.cfg file must be unique. At least one stanza named local must exist. No stanza should be named ldap or default.

**Examples**
* Multiple components of the PKI implementation use this file for configuration information.
* algorithm  Defines the encryption algorithm used for CMP requests.
* Supported values are RSA and DSA. The default is RSA.
* crl  Specifies the CA's root certificate file.
* **dn** Defines the default Distinguished Name value for newly created certificates. (Optional) Example:
  
  \[dn = "c=US, o=ZZZ Corp., ou=Sales OEM, sp=Texas, l=Austin"\]

* **keysize** Defines the minimum number of bits required when generating an encryption/signing key. The default is 1024.

* **program** Specifies the PKI service module file name. (Required)

* **retries** Defines the number of retry attempts when contacting a CA. The default is 5.

* **server** Defines the URL address of the CA server. Example:
  
  \["cmp://9.53.149.39:1077"\]

* **signinghash** Specifies the hash algorithm used to verify keys and to perform trusted certificate signing when validating users. Supported values are MD2, MD5, and SHA1. The default is MD5.

* **trustedkey** Defines the keystore location containing the system-wide trusted signing key used to sign/verify user certificates.

* **url** Defines the default subject alternate name URI value to be added to new certificates.

**local:**
  
  - `program = /usr/lib/security/pki/JSML`
  - `trustedkey = file:/usr/lib/security/pki/trusted.p15`
  - `server = "cmp://9.53.149.39:1077"`
  - `crl = ldap://9.53.149.39/o=XYZ, c=us`
  - `dn = "c=US, o=XYZ"`
  - `url = "http://www.ibm.com/"`
  - `algorithm = RSA`
  - `keysize = 512`
  - `retries = 5`
  - `signinghash = MD5`

**File**

`/usr/lib/security/pki/ca.cfg`

**Related Information**

The `certcreate` command.

The `/usr/lib/security/pki/acct.cfg` file.

---

**cdromd.conf File Format**

**Purpose**

Defines for the `cdromd` daemon the managed devices and supported file system types.

**Description**

The `/etc/cdromd.conf` is the configuration file for the `cdromd` daemon. This file enables you to specify the devices to manage and the file system types to handle.

If you change the `/etc/cdromd.conf` file, run `refresh -s cdromd` or `kill -1 CdromdPID` command to inform the daemon of the changes to its configuration file.
The cdrom daemon reads its configuration file only when it starts, when the cdrom daemon receives a SIGHUP signal, or when the SRC refresh -s cdrom command is entered.

An information line in the cdrom configuration file defines either a device to manage or a file system type to handle. Lines starting with the pound sign (#) are comment lines. Fields in information lines must be separated by spaces or tabs. A device information line starts with <device> keyword and is of the form:

```
device  device_name  mount_point
```

- **device_name**: Contains a valid device name, as printed by the lsdev command, such as:
  ```
  lsdev -Cc cdrom -F name
  ```
- **mount_point**: Contains the path of the directory for the mount operation. It must begin with a /

If there is no line in the configuration file beginning with the device keyword, all the CD-ROM and DVD devices available on the system will be managed by cdromd, and a media inserted in the cd<x> drive will be automatically mounted on /cdrom/cd<x> directory.

A file system type information line starts with the fstype keyword and is of the form:

```
fstype  VfsName  fs_options
```

- **VfsName**: Contains the VFS type used with the -V flag of the mount command. Only cdrfs and udfs types can be used.
- **fs_options**: Contains the comma separated list of options used with the -o flag of the mount command (see mount command man page).

If there is no line beginning with the fstype keyword in the configuration file, the mount command will be called with one of the following options:

- `-V cdrfs -o ro`
- `-V udfs -o ro`

If you want the UDFS file system to be mounted in read/write mode by default, add the following line to the cdromd.conf file:

```
fstype  udfs  rw
```

### Examples

The following example of cdromd.conf file is for a cdrom daemon that:

- Manages cdrom cd0 with inserted media mounted on /mnt with either `-V cdrfs -o ro` or `-V udfs -o ro` options.
- Manages cdrom cd1 with inserted media mounted on /install with either `-V cdrfs -o ro` or `-V udfs -o ro` options.

```
  device  cd0  /mnt
  device  cd1  /install
  fstype  cdrfs  ro
  fstype  udfs  ro
```

### Related Information

The cdmount, cdutil, cdeject, cdumount, cdcheck, mount commands.

The cdrom daemon.
ClientHostName.info File

Purpose
Created by the Network Installation Management (NIM) software to deliver information to the boot environment of NIM client machines.

Note: In AIX Version 4, this is an internal file to the Network Installation Management software and should not be modified manually.

Description
The NIM software creates the ClientHostName.info file to deliver information to the boot environment of NIM client machines. The file resides in the /tftpboot directory on the server of the NIM Shared Product Object Tree (SPOT), with a format of ClientHostName.info where ClientHostName is the hostname of the client machine.

After the client machine performs a network boot, it retrieves a copy of the ClientHostName.info file from the boot server using tftp. The client machine then uses the contents of the ClientHostName.info file to define environment variables for further processing in the boot process.

The ClientHostName.info file is used to support network boot for the following NIM operations:

- Installing the Base Operating System onto standalone machines
- Initializing diskless/dataless machines
- Diagnostics boot

Some of the variables defined in the ClientHostName.info file are common to all operations while others are operation-specific.

The following variables may be defined in the ClientHostName.info file:

Note: These variables are managed by the nim command and should not be modified by other means.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NIM_NAME</td>
<td>Identifies the client machine in the NIM environment.</td>
</tr>
<tr>
<td>NIM_HOSTNAME</td>
<td>Identifies hostname of the client machine.</td>
</tr>
<tr>
<td>NIM_CONFIGURATION</td>
<td>Describes the configuration of the client's resource requirements. Possible values are standalone, diskless, and dataless.</td>
</tr>
<tr>
<td>NIM_MASTER_HOSTNAME</td>
<td>Identifies the hostname of the NIM master in the network.</td>
</tr>
<tr>
<td>NIM_MASTER_PORT</td>
<td>Specifies the port number on the NIM master that should be used for NIM communications.</td>
</tr>
<tr>
<td>RC_CONFIG</td>
<td>Specifies the file that defines the configuration procedures the client machine should follow as it boots. Possible values are rc.bos_inst, rc.dd_boot, and rc.diag.</td>
</tr>
<tr>
<td>NIM_BOSINST_RECOVER</td>
<td>Specifies the script that initializes the BOS installation environment for NIM.</td>
</tr>
<tr>
<td>SPOT</td>
<td>Specifies the location of the Shared Product Object Tree resource that will be used during the boot process.</td>
</tr>
<tr>
<td>ROOT</td>
<td>Specifies the location of the root filesystem that will be mounted by diskless/dataless machines.</td>
</tr>
<tr>
<td>DUMP</td>
<td>Specifies the location of the dump resource that will be mounted by diskless/dataless machines.</td>
</tr>
<tr>
<td>NIM_CUSTOM</td>
<td>Names the command to execute a NIM script during post-installation processing.</td>
</tr>
<tr>
<td>NIM_BOS_IMAGE</td>
<td>Specifies the image from which the Base Operating System will be installed.</td>
</tr>
<tr>
<td>NIM_BOS_FORMAT</td>
<td>Specifies the format of the image that will be used to install the Base Operating System.</td>
</tr>
</tbody>
</table>
Variable | Description
---|---
NIM_HOSTS | Specifies the IP addresses and hostnames of the NIM machines that will participate in the operation.
NIM_MOUNTS | Specifies the filesystems that will be mounted during the operation.
ROUTES | Specifies the routes from the client machine to other networks in the NIM environment. The format of each value is a colon-separated list of the network IP address, the network subnet mask, and the IP address of the gateway to the network.

Example
This example shows the contents of the file `/tftpboot/devon.austin.ibm.com.info` after a bos installation has been enabled via the following command:

```
nim -o bos_inst -a source=rte devon
```

```
export NIM_NAME=devon
export NIM_HOSTNAME=devon.austin.ibm.com
export NIM_CONFIGURATION=standalone
export NIM_MASTER_HOSTNAME=redfish.austin.ibm.com
export NIM_MASTER_PORT=1058
export RC_CONFIG=rc.bos_inst
export NIM_BOSINST_RECOVER="/../SPOT/usr/lpp/bos.sysmgt/nim/methods/c_bosinst_env -a
                          hostname=devon.austin.ibm.com"
export SPOT=redfish.austin.ibm.com:/spot/myspot/usr
export NIM_CUSTOM="/../SPOT/usr/lpp/bos.sysmgt/nim/methods/c_script -a
                   location=redfish.austin.ibm.com:/export/nim/scripts/devon.script"
export NIM_BOS_IMAGE=/SPOT/usr/sys/inst.images/bos
export NIM_BOS_FORMAT=rte
export NIM_HOSTS="129.35.134.9:devon.austin.ibm.com
9.3.84.202:redfish.austin.ibm.com"
export NIM_MOUNTS="redfish.austin.ibm.com:/lppsource/imagedir:/SPOT/usr/sys/inst.images:dir"
export ROUTES="9.3.84.128:255.255.255.128:129.35.128.201"
```

Files

/`tftpboot/ClientHostName.info` | Default location of the `ClientHostName.info` file.

Related Information

The `nim` command.

clsnmp.conf File

Purpose
Contents are used by the `clsnmp` command to identify a host on which an SNMP agent is running.

Description
The contents of the `clsnmp.conf` file used by the `clsnmp` command are as follows. Each entry identifies:
- a host on which an SNMP agent is running,
- the administrative model used to communicate with the host at that agent,
- and the security parameters to be used in the communication.
An entry in the `clsnmp.conf` file has the following syntax:

```
winSnmpName targetAgent admin secName password context secLevel authProto
authKey privProto privKey
```

where:

**winSnmpName**
An administrative name by which the winSNMP code used by `clsnmp` can locate an entry in this configuration file. This value is to be specified on the `-h` keyword for the `clsnmp` command. Valid values are: A character string of 1 to 32 characters. There is no default value.

**targetAgent**
Identification of the target SNMP agent. By default, the port at which the agent is to receive requests is 161. To specify a port other than 161, use the syntax of:

```
host:port_number (host colon port_number)
```

Valid values are a host name of 1 to 80 characters. An IP address Port number, if specified, must be between 1 and 65535. There is no default value.

**admin** Specifies the administrative model supported by the `targetAgent`. Valid values are:

- **snmpv1** Indicates community based security with SNMPv1 message protocol data units.
- **snmpv2c** Indicates community based security with SNMPv2 message protocol data units.
- **snmpv3** Indicates user based security (USM) with SNMPv3 message protocol data units.

There is no default value.

**secName**
Specifies the security name of the principal using this configuration file entry. For user-based security, this is the `userName`. The user must be defined at the `targetAgent`. This field is ignored unless `snmpv3` is specified for the `admin` keyword. Valid values are: A user name of 1 to 32 characters. There is no default value.

**password**
Specifies the password to be used in generating the authentication and privacy keys for this user. If a password is specified, the values of the `authKey` and `privKey` fields will be ignored.

**context**
Specifies the SNMP `contextName` to be used at the target agent. Note, the `contextName` is needed only at agents that support multiple contexts. Otherwise, the only context supported is the null context, which is the default value of this keyword. The CS for OS/390 SNMP agent does not support multiple contexts. This field is ignored unless `snmpv3` is specified for the `admin` keyword. Valid values are: A `contextName` of 1 to 40 32 characters. A '-' (dash) indicates the default. The default value is the null context ("").

**secLevel**
Specifies the security level to be used in communicating with the target SNMP agent when this entry is used. This field is ignored unless `snmpv3` is specified for the `admin` keyword.
Note: Privacy will be supported on CS for OS/390 V2R7 only in a separately orderable FMID. It will not be supported in the base FMID. Valid values are: noAuthNoPriv or none which indicates no authentication or privacy requested. AuthNoPriv or auth indicates authentication is requested, but privacy is not requested. AuthPriv or priv indicates both authentication and privacy are requested (only supported in the additional encryption product). A '-' (dash) indicates the default. The default value is none (noAuthNoPriv).

**authProto**
Specifies the SNMP authentication protocol to be used in communicating with the target SNMP agent when this entry is used. This field is ignored unless snmpv3 is specified for the admin keyword. Valid values are:

HMAC-MD5
Indicates HMAC mode MD5.

HMAC-SHA
Indicates HMAC mode SHA.

A '-' (dash) indicates the default. The default value is no authentication.

**authKey**
Specifies the SNMP authentication key to be used in communicating with the target SNMP agent when this entry is used. This key must be the non-localized key. This field is ignored if the password keyword is used. This field is ignored unless snmpv3 is specified for the admin keyword and a non-default value is specified for authProto. Valid values are:

- A key of 16 bytes (32 hex digits) when authProto is HMAC-MD5
- A key of 20 bytes (40 hex digits) when authProto is HMAC-SHA

A '-' (dash) indicates the default. The default value is no key.

**privProto**
Specifies the SNMP privacy protocol to be used in communicating with the target SNMP agent when this entry is used.

Note: Privacy will be supported on CS for OS/390 V2R7 only in a separately orderable FMID. It will not be supported in the base FMID. If privacy is not supported, this keyword will be ignored. This field is ignored unless snmpv3 is specified for the admin keyword. Valid values are:

- DES - for CBC-DES (only supported in the additional encryption product)

A '-' (dash) indicates the default. The default value is no privacy.

**privKey**
Specifies the SNMP privacy key to be used in communicating with the target SNMP agent when this entry is used. This key must be the non-localized key. If privacy is not supported, this keyword will be ignored. The privacy and authentication keys are assumed to have been generated using the same authentication protocol (e.g., both with HMAC-MD5 or both with HMAC-SHA). This field is ignored unless snmpv3 is specified for the admin keyword and a non-default value is specified for privProto. Valid values are:

- A key of 16 bytes (32 hex digits) when authProto is HMAC-MD5
- A key of 20 bytes (40 hex digits) when authProto is HMAC-SHA

A '-' (dash) indicates the default. The default value is no key.

**General Usage Rules**
- All parameters for an entry must be contained on one line in the # configuration file.
- A "-" (dash) is used to indicate the default value for a keyword.
- Sequence numbers are not allowed on the statements.
• Comments may be included in the file beginning with a pound sign (#) in column 1.
• The secName and password parameters are case-sensitive.

As the clsnmp command supports both issuance of SNMP requests and receipt of SNMP traps, the entries in the clsnmp.conf file must be defined for both uses. Multiple entries for the same USM user are allowed within the file. This may be useful to define different security levels for the same user. If multiple entries for the same USM user are defined, be aware that only the first one in the file can be used for receiving notifications. If multiple entries for the same USM user are defined and the user will be used for receiving notifications, the definition with the highest (most stringent) securityLevel should be defined first. Doing so will allow the user to be used for any level of security equal to or lower (less stringent) than the securityLevel defined.

Related Information
The snmpdv3 clsnmp pwtokey and pwchange commands.
The /etc/clsnmp.conf file.
The snmpdv3.conf file.

Problem Determination for the SNMP Daemon, SNMP trap processing, SNMP daemon troubleshooting in Networks and communication management.

SNMP daemon configuration in Networks and communication management.

Command (C.*) Files for BNU

Purpose
Contains file transfer directions for the uucico daemon.

Description
Command (C.*) files contain the directions that the Basic Networking Utilities (BNU) uucico daemon follows when transferring files. The full path name of a command file is a form of the following:

/var/spool/uucp/SystemName/C.SystemNameNxxxx

The SystemName variable indicates the name of the remote system. The N character represents the grade of the work. The xxxx notation is the four-digit hexadecimal transfer-sequence number; for example, C.merlinC3119.

The grade of the work specifies when the file is to be transmitted during a particular connection. The grade notation characteristics are:
• A single number (0-9) or letter (A-Z, a-z)
• Lower sequence characters cause the file to be transmitted earlier in the connection than do higher sequence characters. Sequence is established using ASCII order, beginning with 0 and ending with z.
• The number 0 is the highest grade (that is, the lowest character in the sequence), signifying the earliest transmittal; z is the lowest grade, specifying the latest transmittal.
• The default grade is N.

A command file consists of a single line that includes the following kinds of information in the following order:
1. An S (send) or R (receive) notation.
A send command file is created by the `uucp` or `uuto` commands; a receive command file is created by the `uux` command.

2. The full path name of the source file being transferred. A receive command file does not include this entry.

3. The full path name of the destination file, or a path name preceded by `~user`, where `user` is a login name on the specified system. Here, the `~` (tilde) is shorthand for the name of the user's home directory.

4. The sender's login name.

5. A list of the options, if any, included with the `uucp`, `uuto`, or `uux` command.

6. The name of the data file associated with the command file in the spooling directory. This field must contain an entry. If one of the data-transfer commands (such as the `uucp` command with the default `-c` flag) does not create a data file, the BNU program instead creates a placeholder with the name `D.0` for send files or the name `dummy` for receive files.

7. The source file permissions code, specified as a three-digit octal number (for example, 777).

8. The login name of the user on the remote system who is to be notified when the transfer is complete.

**Examples**

The following are two examples of using the command (C.*) files.

**Two Send Command Files**

1. The send command file `/var/spool/uucp/venus/C.heraN1133`, created with the `uucp` command, contains the following fields:

   \[ \text{S } /home/amy/f1 /var/spool/uucppublic/f2 amy -d C.hera73655 777 lgh } \]
   
   where:
   
   a. `S` denotes that the `uucp` command is sending the file.
   
   b. The full path name of the source file is `/home/amy/f1`.
   
   c. The full path name of the destination is `/var/spool/uucppublic/f2`, where `/var/spool/uucppublic` is the name of the BNU public spooling directory on the remote computer and `f2` is the new name of the file.
   
   **Note:** The destination name may be abbreviated as `~/f2`. Here, the `~` (tilde) is a shorthand way of designating the public directory.
   
   d. The person sending the file is `amy`.
   
   e. The sender entered the `uucp` command with the `-C` flag, specifying that the `uucp` command program should transfer the file to the local spooling directory and create a data file for it. (The `-d` flag, which specifies that the command should create any intermediate directories needed to copy the source file to the destination, is a default.)
   
   f. The name of the data (D.*) file is `D.hera73655`, which the `uucp` command assigns.
   
   g. The octal permissions code is 777.
   
   h. The `lgh` login name of the user on system `hera`, who is to be notified of the file arrival.

2. The `/var/spool/uucp/hera/C.zeusN3130` send command file, produced by the `uuto` command, is as follows:

   \[ \text{S } /home/amy/out */receive/msg/zeus amy -dcn D.0 777 msg } \]
   
   The `S` denotes that the `/home/amy/out` source file was sent to the `receive/msg` subdirectory in the public spooling directory on system `zeus` by user `amy`.
   
   **Note:** The `uuto` command creates the `receive/msg` directory if it does not already exist.

   The `uuto` command used the default flags `-d` (create directories), `-c` (transfer directly, no spooling directory or data file), and `-n` (notify recipient). The `D.0` notation is a placeholder, 777 is the permissions code, and `msg` is the recipient.
Receive Command File

The format of a receive command file is somewhat different from that of a send command file. When files required to run a specified command on a remote system are not present on that system, the **uux** command creates a receive command file.

For example, the following command:
```
uux - "diff /home/amy/out heral/home/amy/out2 > "~/DF"
```

produces the `/var/spool/uucp/zeus/C.heraR1e94` receive command file.

**Note:** The command in this example invokes the **uux** command to run a **diff** command on the local system, comparing file `/home/amy/out` with file `/home/amy/out2`, which is stored on the remote system **hera**. The output of the comparison is placed in the `DF` file in the public directory on the local system.

The actual receive command file looks like this:
```
R /home/amy/out2 D.hera1e954fd amy - dummy 0666 amy
```

The `R` denotes a receive file. The **uucico** daemon, called by the **uux** command, gets the `/home/amy/out2` file from system **hera** and places it in a data file called `D.hera1e954fd` for the transfer. Once the files are transferred, the **uuxqt** daemon executes the command on the specified system.

User **amy** issued the **uux** command with the `-` (minus sign) flag, which makes the standard input to the **uux** command the standard input to the actual command string. No data file was created in the local spooling directory, so the BNU program uses `dummy` as a placeholder. The permissions code is `666` (the BNU program prefixes the three-digit octal code with a `0`), and user **amy** is to be notified when the command has finished executing.

**Files**

- `/etc/uucp/Permissions` - Describes access permissions for remote systems.
- `/etc/uucp/Systems` - Describes accessible remote systems.
- `/etc/uucp/Sysfiles` - Specifies possible alternative files for `/etc/uucp/Systems`.
- `/var/spool/uucp/` `SystemName` `/D.*` - Contains data to be transferred.
- `/var/spool/uucp/` `SystemName` directory - Contains BNU command, data, and execute files.
- `/var/spool/uucppublic/*` directory - Contains transferred files.

**Related Information**

The **uucp** command, **uudemon.cleanu** command, **uupick** command, **uto** command, **uux** command.

The **cron** daemon, **uucico** daemon, **uusched** daemon, **uuxqt** daemon.

[BNU File and Directory Structure] [BNU Daemons] [BNU maintenance commands] in *Networks and communication management*.

**compver File**

**Purpose**

Describes the format of a compatible versions file.
Description
The compver file is an ASCII file used to specify previous versions of the associated package which are upward compatible. It is created by a package developer.

Each line of the file specifies a previous version of the associated package with which the current version is backward compatible.

Since some packages may require installation of a specific version of another software package, compatibility information is extremely crucial. Consider, for example, a package called “A” which requires version “1.0” of application “B” as a prerequisite for installation. If the customer installing “A” has a newer version of “B” (1.3), the compver file for “B” must indicate that “1.3” is compatible with version “1.0” in order for the customer to install package “A.”

The comparison of the version string disregards white space and tabs. It is performed on a word-by-word basis. Thus Version 1.3 and Version 1.3 would be considered the same.

Examples
An example of a compver file is shown below.

    Version 1.3
    Version 1.0

Related Information
The depend file format.

cfg File

Purpose
Contains audit system configuration information.

Description
The /etc/security/audit/config file is an ASCII stanza file that contains audit system configuration information. This file contains five stanzas: start, bin, stream, classes, and users.

start Stanza
The start stanza contains the attributes used by the audit start command to initialize the audit system. The format follows:

    start:
      binmode = off | on | panic
      streammode = off | on

The attributes are defined as follows:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>binmode</td>
<td>Controls whether bin collection, as defined in the bin stanza, is used.</td>
</tr>
<tr>
<td></td>
<td>off</td>
</tr>
<tr>
<td></td>
<td>on</td>
</tr>
<tr>
<td></td>
<td>panic</td>
</tr>
<tr>
<td>Attribute</td>
<td>Definition</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>streammode</td>
<td>Controls whether stream data collection, as defined in the file specified in the stream stanza (normally the <code>/etc/security/audit/streamcmds</code> file), is configured at the start up of the audit system.</td>
</tr>
<tr>
<td>off</td>
<td>Stream data collection is not enabled. This is the default value.</td>
</tr>
<tr>
<td>on</td>
<td>Stream data collection is enabled.</td>
</tr>
</tbody>
</table>

**Note:** If neither collection mode is defined or if both modes are in the off state, only subsystem configuration is done.

**bin Stanza**

The **bin** stanza contains the attributes used by the **auditbin** daemon to set up bin mode auditing. The format follows:

```
bin:
  trail = PathName
  bin1 = PathName
  bin2 = PathName
  binsize = DecimalString
  cmds = PathName
  bytethreshold = DecimalString
  eventthreshold = DecimalString
  freespace = DecimalString
```

Bin mode parameters are defined as follows:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>trail</td>
<td>Specifies the path name of the audit trail file. When this is defined, the <strong>auditbin</strong> daemon can substitute the path name of the audit trail file for the <strong>$trail</strong> string in the backend commands that it calls.</td>
</tr>
<tr>
<td>bin1</td>
<td>Specifies the path name that the <strong>auditbin</strong> daemon uses for its primary bin file. If the <strong>$bin</strong> string is the parameter value, the <strong>auditbin</strong> daemon substitutes the name of the current bin file.</td>
</tr>
<tr>
<td>bin2</td>
<td>Specifies the path name that the <strong>auditbin</strong> daemon uses for its secondary bin file. If the <strong>$bin</strong> string is the parameter value, the <strong>auditbin</strong> daemon substitutes the name of the current bin file.</td>
</tr>
<tr>
<td>binsize</td>
<td>Specifies a decimal integer string that defines the threshold size (in bytes) of each audit bin. If the <strong>binsize</strong> parameter is set to 0, no bin switching will occur, and all bin collection will go to <strong>bin1</strong>.</td>
</tr>
<tr>
<td>cmds</td>
<td>Specifies the path name of the file that contains the audit backend commands called by the <strong>auditbin</strong> daemon. The file contains command lines, each composed of one or more backend commands with input and output that can be piped together or redirected. See the description of the <code>/etc/security/audit/bincmds</code> file for more information.</td>
</tr>
<tr>
<td>bytethreshold</td>
<td>Specifies the decimal integer string that defines the approximate number of bytes written to an audit bin before a synchronous update is performed. If the <strong>bytethreshold</strong> is set to 0, this function is disabled. Both <strong>bytethreshold</strong> and <strong>eventthreshold</strong> can be used simultaneously.</td>
</tr>
<tr>
<td>eventthreshold</td>
<td>Specifies a decimal integer string that defines the maximum number of events written to an audit bin before a synchronous update is performed. If the <strong>eventthreshold</strong> is set to 0, this function is disabled. Both <strong>eventthreshold</strong> and <strong>bytethreshold</strong> can be used simultaneously.</td>
</tr>
<tr>
<td>freespace</td>
<td>Specifies a decimal integer string that defines the recommended number of 512-byte free blocks in the file system where the audit trail file is located. If the free space of file system is below this value, audit generates a warning message through the <strong>syslog</strong> subsystem every time that the audit bin is switched. The default value is 65536 blocks (64 megabytes). The maximum possible value is 4194303 (about 2GB of free disk space). If this value is set to 0, no warning message is generated.</td>
</tr>
</tbody>
</table>
stream Stanza

The stream stanza contains the attributes that the **audit start** command uses to set up initial stream mode auditing. The format follows:

```plaintext
cmds = PathName
```

The `PathName` parameter identifies the file that contains the stream commands that are executed at the initialization of the audit system. These commands can use shell piping and redirection, but no substitution of path names is performed on `$trail` or `$bin` strings.

classes Stanza

The classes stanza defines audit classes (sets of audit events) to the system.

Each audit class name must be less than 16 characters and be unique on the system. Each class definition must be contained in a single line, with a new line acting as a delimiter between classes. The system supports up to 32 audit classes, with ALL as the last class. The audit events in the class must be defined in the `/etc/security/audit/events` file.

```plaintext
classes:
    auditclass = auditevent, ...auditevent
```

users Stanza

The users stanza defines audit classes (sets of events) for each user. The classes are defined to the operating system kernel.

The format is as follows:

```plaintext
users:
    UserName = auditclass, ... auditclass
```

Each `UserName` attribute must be the login name of a system user or the string `default`, and each `auditclass` parameter should be defined in the **classes** stanza.

To establish the audit activities for a user, use the **chuser** command with the **auditclasses** attribute.

Security

Access Control: This file should grant read (r) access to the root user and members of the audit group and write (w) access only to the root user.

<table>
<thead>
<tr>
<th>Event</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUD_CONFIG_WR</td>
<td>file name</td>
</tr>
</tbody>
</table>

Examples

1. To define audit classes, add a line to the **classes** stanza of the `/etc/security/audit/config` file for each set of events that you want to assign to a class:

```plaintext
classes:
    general = USER_SU,PASSWORD_Change,FILE_Unlink,
              FILE_Link,FILE_Remove
    system = USER_Change,GROUP_Change,USER_Create,
             GROUP_Create
    init = USER_Login, USER_Logout
```

   These specific audit events and audit classes are described in “Setting Up Auditing” in Operating system and device management.

2. To establish the audit activities for each user, use the **chuser** command with the **auditclasses** attribute for each user for whom you want to define audit classes (sets of audit events):
chuser "auditclasses=general,init,system" dave
chuser "auditclasses=general,init" mary

These chuser commands create the following lines in the users stanza of the /etc/security/audit/config file:

users:
dave=general,init,system
mary=general,init

This configuration includes dave, the administrator of the system, and mary, an employee who updates information.

3. To enable the auditing system, turn on bin data collection, and turn off initial stream data collection, add the following to the start stanza of the /etc/security/audit/config file:

   start:
   binmode = on
   streammode = off

4. To enable the auditbin daemon to set up bin collection, add attributes to the bin stanza of the /etc/security/audit/config file:

   bin:
   trail = /audit/trail
   bin1 = /audit/bin1
   bin2 = /audit/bin2
   binsize = 25000
   cmds = /etc/security/audit/bin cmds

   The attribute values in the preceding stanza enable the audit system to collect bin files of data and store the records in a long-term audit trail.

5. To enable the auditbin daemon to set up stream collection, add lines to the start and stream stanzas of the /etc/security/audit/config file:

   start:
   streammode = on
   stream:
   cmds = /etc/security/audit/streamcmds

Files

/etc/security/audit/config Specifies the path to the file.
/etc/security/audit/objects Contains audit events for audited objects.
/etc/security/audit/events Contains the audit events of the system.
/etc/security/audit/bin cmds Contains auditbin backend commands.
/etc/security/audit/streamcmds Contains auditstream commands.

Related Information
The audit command, auditbin daemon, chuser command.
The auditproc subroutine.

Setting Up Auditing in Operating system and device management.

Security Administration Auditing overview in Operating system and device management.
consdef File

Purpose
Enables asynchronous tty devices to be console candidates at system boot when no console device is defined or available.

Description
The `/etc/consdef` file enables tty devices such as terminals and modems to be chosen as the console device. When the console device is undefined, the system displays a message on all natively attached graphics displays and the tty on native serial port S1. The console device is undefined when:

- The system is first installed and started.
- The console definition has been deleted from the ODM database.
- The console device has been physically removed from the system.

If any of these conditions occur, the system displays the following message:

```
******* Please define the System Console. *******
```

Type a Number and press <Enter> to use this terminal as the system console.

For high function terminals (HFTs) graphics displays, the Number variable refers to a function key. For asynchronous ttys, this variable is a number.

The selected item becomes the system console. To choose a non-default tty device as the system console, you must first configure the `/etc/consdef` file. This file contains stanzas that define various console attributes. Each line, or entry, in a stanza must take the form of `Attribute=Value`, and the line must not exceed 80 characters. The following attributes must be defined for each terminal device:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>connection</td>
<td>Identifies the type of tty interface. Valid values are rs232 and rs422.</td>
</tr>
<tr>
<td>location</td>
<td>Specifies the location code of the terminal. Location codes of 00-00-S1-00 or 00-00-S2-00 indicate that the tty device is attached to the S1 or S2 serial port, respectively. Any other location code indicates the tty device is attached to an adapter card other than the standard I/O planar. You can display valid location values with the `lsdev -C</td>
</tr>
</tbody>
</table>

You can also specify other terminal attributes such as `speed`, `bpc`, `stops`, `parity`, and `term`. If you do not define these attributes, the system uses the default values stored in the ODM database. The `consdef` file contains a sample stanza for the S1 port. To enable this stanza, or parts of it, remove the comment delimiters (#) from each applicable line.

Examples
To display the console selection message on the ttys attached to the S1 and S2 ports:

```
ALTTTY:
    connection=rs232
    location=00-00-S1-00
    speed=9600
    bpc=8
    stops=1
    parity=none
    term=ibm3163

ALTTTY:
    connection=rs232
    location=00-00-S2-00
    speed=9600
```
Note: For backward compatibility, the ALTTY: keyword is not required for the first entry.

Files

/etc/consdef
/dev/console

Specifies the path of the consdef file.

Provides access to the system console.

Related Information

The chcons command.

The lsdev command.

The console special file.

Device location codes in Operating system and device management.

copyright File

Purpose

Describes the format of a copyright information file.

Description

The copyright file is an ASCII file used to provide a copyright notice for a package. The text may be in any format. The full file contents (including comment lines) is displayed on the terminal at the time of package installation.

copyright File

Purpose

Default location for the local node’s cluster security services public key file.

Description

The /var/ct/cfg/ct_has.pkf file is the default location where the ctcasd demon will expect to find the local node’s public key file. The public key is stored in a proprietary binary format.

The ctcasd.cfg file permits the system administrator to specify an alternate location for this file. The ctskeygen -p command permits the administrator to create this file in an alternate location. If an alternate location is used, the file must meet all the criteria listed in the Security section of this man page. The file must not be recorded to a read-only file system, because this will prohibit the system administrator for modifying the contents of this file in the future.

If the ctcasd daemon cannot locate this file during its startup, it will check for the presence of the ct_has.qkf file. If both files are missing, the daemon assumes that it is being started for the first time after installation, and create an initial private and public key file for the node. The daemon also creates the initial trusted host list file for this node. This file contains an entry for localhost and the host names (or IP addresses) associated with all AF_INET-configured adapters that the daemon can detect. This may cause inadvertent authentication failures if the public and private key files were accidentally or intentionally
removed from the local system before the daemon was restarted. **ctcasd** will create new keys for the node, which will not match the keys stored on the other cluster nodes. If UNIX-identity-based authentication suddenly fails after a system restart, this is a possible source of the failure.

If the public key file is missing but the private key file is detected, the daemon concludes that the local node is misconfigured and terminates. A record is made to persistent storage to indicate the source of the failure.

**Security**

This file is readable to all users on the local system. Write permission is not granted to any system user.

By default, this file is stored in a locally-mounted file system. The **ctcasd.cfg** file permits system administrators to change the location of the file. Should system administrators use a different location, it is the administrator's responsibility to assure that the file is always accessible to the local node, and that all users from this local node can read the file. If the storage location does not meet these criteria, users and applications will be unable to authenticate to trusted services using UNIX-identity-based authentication.

If the system administrator chooses to place this file in a networked file system, the administrator must assure that no two nodes are attempting to use the same physical file as their own public key file. Because public keys differ between nodes, if two nodes attempt to use the same public key file, at least one of them will always obtain the incorrect value for its public key. This will cause applications and users from that node to fail authentication to trusted services within the cluster.

**Restrictions**

Cluster security services supports only its own private and public key formats and file formats. Secured Remote Shell formats are currently unsupported. Settings for the **HBA_USING_SSH_KEYS** attribute are ignored.

**Examples**

This example shows the default contents of the configuration file:

```
TRACE= ON
TRACEFILE=/var/ct/IW/log/ctsec/ctcasd/trace
TRACELEVELS= _SEC:Info=1,_SEC:Errors=1
TRACESIZE= 1003520
RQUEUESIZE= MAXTHREADS= MINTHREADS= THREADSTACK= 131072
HBA_USING_SSH_KEYS= false
HBA_PRVKEYFILE= HBA_PUBKEYFILE= HBA_THLFILE= HBA_KEYGEN_METHOD= rsa512
SERVICES=hba CAS
```

After modification, the contents of the configuration file might look like this:

```
TRACE= ON
TRACEFILE=/var/ct/IW/log/ctsec/ctcasd/trace
TRACELEVELS= _SEC:Perf=1,_SEC:Errors=8
TRACESIZE= 1003520
  RQUEUESIZE= 64
  MAXTHREADS= 10
  MINTHREADS= 4
  THREADSTACK= 131072
HBA_USING_SSH_KEYS= false
  HBA_PRVKEYFILE= /var/ct/cfg/qkey
```

28 AIX 5L Version 5.3 Files Reference
HBA_PUBKEYFILE= /var/ct/cfg/pkey  
HBA_THLFILE= /var/ct/cfg/thl  
HBA_KEYGEN_METHOD= rsa512  
SERVICES= hba CAS

Location
/var/ct/cfg/ct_has.pkf Contains the ct_has.pkf file

Files
/usr/sbin/rsct/cfg/ctcasd.cfg Default location of the ctcasd.cfg file

Related Information
Commands: ctskeygen  
Daemons: ctcasd  
Files: ct_has.qkf, ct_has.thl

ct_has.qkf File

Purpose
Default location for the cluster security services private key file for the local node.

Description
The /var/ct/cfg/ct_has.qkf file is the default location where the ctcasd demon will expect to find the local node’s private key file. The private key is stored in a proprietary binary format.

The ctcasd.cfg file permits the system administrator to specify an alternate location for this file. The ctskeygen -q command permits the administrator to create this file in an alternate location. If an alternate location is used, the file must meet all the criteria listed in the Security section of this man page. The file must not be recorded to a read-only file system, because this will prohibit the system administrator for modifying the contents of this file in the future.

If the ctcasd daemon cannot locate this file during its startup, it will check for the presence of the ct_has.pkf file. If both files are missing, the daemon will assume that it is being started for the first time after installation, and create an initial private and public key file for the node. The daemon also creates the initial trusted host list file for this node. This file contains an entry for localhost and the host names (or IP addresses) associated with all AF_INET-configured adapters that the daemon can detect. This may cause inadvertent authentication failures if the public and private key files were accidentally or intentionally removed from the local system before the daemon was restarted, ctcasd will create new keys for the node, which will not match the keys stored on the other cluster nodes. If UNIX-identity-based authentication suddenly fails after a system restart, this is a possible source of the failure.

If the private key file is missing but the public key file is detected, the daemon concludes that the local node is misconfigured and terminates. A record is made to persistent storage to indicate the source of the failure.

Security
This file is readable and accessible only to the root user. Access to all other users is not provided.

By default, this file is stored in a locally mounted file system. The ctcasd.cfg file permits system administrators to change the location of the file. Should system administrators use a different location, it is
the administrator’s responsibility to assure that the file is always accessible to the local node, and that only the root user from this local node can access the file. If the storage location does not meet these criteria, the security of the node and the cluster should be considered compromised.

Restrictions
Cluster security services supports only its own private and public key formats and file formats. Secured Remote Shell formats are currently unsupported. Settings for the HBA_USING_SSH_KEYS attribute are ignored.

Examples
This example shows the default contents of the configuration file:

```
TRACE= ON
TRACEFILE= /var/ct/IW/log/ctsec/ctcasd/trace
TRACELEVELS= _SEC:Info=1,_SEC:Errors=1
TRACESIZE= 1003520
RQUEUESIZE= MAXTHREADS= MINTHREADS= THREADSTACK= 131072
HBA_USING_SSH_KEYS= false
HBA_PRIVKEYFILE= HBA_PUBKEYFILE= HBA_THLFILE= HBA_KEYGEN_METHOD= rsa512
SERVICES=hba CAS
```

After modification, the contents of the configuration file might look like this:

```
TRACE= ON
TRACEFILE= /var/ct/IW/log/ctsec/ctcasd/trace
TRACELEVELS= _SEC:Perf=1,_SEC:Errors=8
TRACESIZE= 1003520
    RQUEUESIZE= 64
    MAXTHREADS= 10
    MINTHREADS= 4
    THREADSTACK= 131072
HBA_USING_SSH_KEYS= false
    HBA_PRIVKEYFILE= /var/ct/cfg/qkey
    HBA_PUBKEYFILE= /var/ct/cfg/pkey
    HBA_THLFILE= /var/ct/cfg/thl
    HBA_KEYGEN_METHOD= rsa512
SERVICES= hba CAS
```

Location
```
/usr/sbin/rsct/bin/ct_has.qkf   Location of the ct_has.qkf file.
```

Files
```
/usr/sbin/rsct/cfg/ctcasd.cfg   Default location of the ctcasd.cfg file
```

Related Information
Commands: ctskeygen

Daemons: ctcasd

Files: ct_has.pkf, ct_has.thl

Books: see the RSCT Administration Guide for information about the event response resource manager
ct_has.thl File

Purpose
Default location for the local node’s cluster security services trusted host list file.

Description
The /var/ct/cfg/ct_has.thl file is the default location where the ctcasd daemon will expect to find the local node’s trusted host list file. The contents of this file are stored in a proprietary binary format.

The trusted host list maps each host identity within the peer domain or management domain to the host’s cluster security services public key. The ctcasd daemon uses this list to determine which nodes on the network are trusted, and to locate the public keys for these nodes in order to decrypt UNIX-identity-based credentials transmitted from another host within the cluster. If a host is not listed in a node’s trusted host list, or if the public key recorded for that host is incorrect, the host will not be able to authenticate to that node using UNIX-identity-based authentication.

The ctcasd.cfg file permits the system administrator to specify an alternate location for this file. If an alternate location is used, the file must meet all the criteria listed in the Security section of this man page. The file must not be recorded to a read-only file system, because this will prohibit the system administrator for modifying the contents of this file in the future.

If the ctcasd daemon cannot locate this file during its startup, it will check for the presence of the ct_has.pkf file. If both files are missing, the daemon will assume that it is being started for the first time after installation, and create an initial private and public key file for the node. The daemon also creates the initial trusted host list file for this node. This file contains an entry for localhost, along with the IP addresses and the host names associated with all AF_INET-configured adapters that the daemon can detect. This may cause inadvertent authentication failures if the public and private key files were accidentally or intentionally removed from the local system before the daemon was restarted. ctcasd will create new keys for the node, which will not match the keys stored on the other cluster nodes. If UNIX-identity-based authentication suddenly fails after a system restart, this is a possible source of the failure.

Security
This file is readable by all users on the local system. Write access is not provided to any system user.

By default, this file is stored in a locally-mounted file system. The ctcasd.cfg file permits system administrators to change the location of the file. If the system administrator uses a different location, it is the administrator’s responsibility to make sure the file is always accessible to the local node, and that all users from this local node can access the file. If the storage location does not meet these criteria, users and applications will be unable to authenticate to trusted services using UNIX-identity-based authentication.

If the system administrator chooses to place this file in a networked file system, the administrator must assure that no two nodes are attempting to use the same physical file as their own trusted host list file, or that the file does not contain an entry for localhost. By default, the trusted host list contains an entry for localhost, which maps the local system’s public key to this value. If multiple hosts share the same trusted host list file, attempts by users or applications to contact localhost for trusted services may fail because the entry maps to an incorrect public key value.

Restrictions
• Cluster security services supports only its own private and public key formats and file formats.
• Cluster security services does not provide an automated utility for creating, managing, and maintaining trusted host lists throughout the cluster. This is a procedure left to either the system administrator or the cluster management software.

Examples
This example shows the default contents of the configuration file:

```
TRACE=ON
TRACEFILE=/var/ct/IW/log/ctsec/ctcasd/trace
TRACELEVELS=_SEC:Info=1,_SEC:Errors=1
TRACESIZE=1003520
RQUEUESIZE=
MAXTHREADS=
MINTHREADS=
THREADSTACK=131072
HBA_USING_SSH_KEYS=false
HBA_PVKEYFILE=
HBA_PUBKEYFILE=
HBA_THLFILE=
HBA_KEYGEN_METHOD=rsa512
SERVICES=hba CAS
```

After modification, the contents of the configuration file might look like this:

```
TRACE=ON
TRACEFILE=/var/ct/IW/log/ctsec/ctcasd/trace
TRACELEVELS=_SEC:Perf=1,_SEC:Errors=8
TRACESIZE=1003520
RQUEUESIZE=64
MAXTHREADS=10
MINTHREADS=4
THREADSTACK=131072
HBA_USING_SSH_KEYS=false
HBA_PVKEYFILE=/var/ct/cfg/qkey
HBA_PUBKEYFILE=/var/ct/cfg/pkey
HBA_THLFILE=/var/ct/cfg/thl
HBA_KEYGEN_METHOD=rsa512
SERVICES=hba CAS
```

Location
/usr/sbin/rsct/bin/ct_has.thl Location of the ct_has.thl file.

Files
/usr/sbin/rsct/cfg/ctcasd.cfg Default location of the ctcasd.cfg file

Related Information
Commands: ctkeygen, cttshl

Daemons: ctcasd

Files: ct_has.pkf, ct_has.qkf

ctcasd.cfg File

Purpose
Provides operational parameters to the cluster security services daemon ctcasd.
Description

The ctcasd.cfg configuration file defines the operational parameters to the cluster security services daemon ctcasd. The ctcasd daemon reads this file when it (the daemon) initializes. The ctcasd daemon expects to find this configuration file in either the /var/ct/cfg directory (preferred) or in the /usr/sbin/rsctcfg directory (default). System administrators can modify the contents of the file stored in the /var/ct/cfg directory, but should not modify the default version of the file in /usr/sbin/rsctcfg unless instructed to do so by the cluster software service provider.

This file is ASCII-formatted, and can be modified using any available text editor. One attribute can be defined per line within this file. Attributes are specified as follows:

attribute=value

The following attributes are defined:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRACE</td>
<td>Indicates whether daemon tracing is activated. Acceptable values are ON and OFF. If the TRACE attribute is not listed in the ctcasd.cfg file, tracing is not activated. For coexistence with earlier versions of RSCT, TRACE=false is interpreted as TRACE=OFF.</td>
</tr>
<tr>
<td>TRACEFILE</td>
<td>Specifies the fully-qualified path name where daemon tracing information is to be recorded.</td>
</tr>
<tr>
<td>TRACELEVELS</td>
<td>Indicates the tracing granularity employed by the daemon when tracing is activated. The possible trace categories are:</td>
</tr>
<tr>
<td></td>
<td>_SEC:Errors Captures error information in the trace log. Possible values are: 1, 2, 4, and 8.</td>
</tr>
<tr>
<td></td>
<td>_SEC:API Tracks the entry and exit of subroutines within the daemon. Possible values are: 1 and 8.</td>
</tr>
<tr>
<td></td>
<td>_SEC:Perf Captures performance-related information. Possible values are: 1, 4, and 8.</td>
</tr>
<tr>
<td></td>
<td>_SEC:Info Traces the general execution progress of the daemon. Possible values are: 1, 2, 3, 4, and 7.</td>
</tr>
</tbody>
</table>

When setting the values of these trace categories, keep in mind that the lower the number is, the less intrusive (and less detailed) the trace will be. Multiple traces can be enabled at once. For example, if an administrator wants to enable a trace that captures basic performance data and highly-detailed error data, the specification for TRACELEVELS would be:

TRACELEVELS=_SEC:Perf=1,_SEC:Errors=8

TRACESIZE Specifies the size of the trace file in bytes. The default value is 1 megabyte.

RQUEUESIZE Indicates the maximum length permitted for the daemon’s internal run queue. If this value is not set, a default value of 64 is used.

MAXTHREADS The limit to the number of working threads that the daemon may create and use at any given time (the "high water mark"). If this value is not set, a default value of 10 is used.

MINTHREADS The number of idle threads that the daemon will retain if the daemon is awaiting further work (the "low water mark"). If this value is not set, a default value of 4 is used.
THREADSTACK
Sets the internal memory used by the daemon for thread stack space. The value is expressed in bytes. If no value is specified, the default system thread stack size is used. This value should not be modified by the administrator unless instructed to do so by IBM Service.

HBA_USING_SSH_KEYS
Indicates whether the daemon is making use of Secured Remote Shell keys. Acceptable values are true and false. If this value is not defined, a default value of false is used. See Restrictions.

HBA_PRVKEYFILE
Provides the full path name of the file that contains the local node’s private key. If this value is not set, the default location of /var/ct_cfg/ct_has.qkf is used.

HBA_PUBKEYFILE
Provides the full path name of the file that contains the local node’s public key. If this value is not set, the default location of /var/ct_cfg/ct_has.pkf is used.

HBA_THLFILE
Provides the full path name of the file that contains the local node’s trusted host list. If this value is not set, the default location of /var/ct_cfg/ct_has.thl is used.

HBA_KEYGEN_METHOD
Indicates the method to be used by ctcasd to generate the private and public keys of the local node if the files containing these keys do not exist. Acceptable values are those that can be provided as arguments to the ctskeygen -m command. If no value is provided for this attribute, the default value of rsa1024 is used.

SERVICES
Lists the internal cluster security services library services that the daemon supports. This entry should not be modified by system administrators unless they are explicitly instructed to do so by the cluster security software service provider.

Restrictions
Cluster security services supports only its own private and public key formats and file formats. Secured Remote Shell formats are currently unsupported. Settings for the HBA_USING_SSH_KEYS attribute are ignored.

Examples
This example shows the default contents of the configuration file:

TRACE= ON
TRACEFILE= /var/ct/IW/log/ctsec/ctcasd/trace
TRACELEVELS= _SEC:Info=1,_SEC:Errors=1
TRACESIZE= 1003520
RQUEUESIZE=
MAXTHREADS=
MINTHREADS=
THREADSTACK= 131072
HBA_USING_SSH_KEYS= false
HBA_PRVKEYFILE=
HBA_PUBKEYFILE=
HBA_THLFILE=
HBA_KEYGEN_METHOD= rsa512
SERVICES= hba CAS

After modification, the contents of the configuration file might look like this:

TRACE= ON
TRACEFILE= /var/ct/IW/log/ctsec/ctcasd/trace
TRACELEVELS= _SEC:Perf=1,_SEC:Errors=8
TRACESIZE= 1003520
RQUEUESIZE= 64
MAXTHREADS= 10
MINTHREADS= 4
THREADSTACK= 131072
HBA_USING_SSH_KEYS= false
   HBA_PVTKEYFILE= /var/ct/cfg/qkey
   HBA_PUBKEYFILE= /var/ct/cfg/pkey
   HBA_THLFILE= /var/ct/cfg/thl
   HBA_KEYGEN_METHOD= rsa512
SERVICES= hba CAS

**Location**

/var/ct/cfg/ctcasd.cfg Contains the ctcasd.cfg file

**Files**

/usr/sbin/rsct/cfg/ctcasd.cfg Default location of the ctcasd.cfg file

**Related Information**

Commands: ctskeygen

Daemons: ctcasd

Files: ct_has.pkf, ct_has.qkf

---

### ctrmc.acls File

**Purpose**

Contains a node’s resource monitoring and control (RMC) access control list (ACL).

**Description**

RMC implements authorization using an access control list (ACL) file. Specifically, RMC uses the ACL file on a particular node to determine the permissions that a user must have in order to access resource classes and their resource instances. A node’s RMC ACL file is named `ctrmc.acls` and is installed in the directory `/usr/sbin/rsct/cfg`. You can allow RMC to use the default permissions set in this file, or you can modify the file after copying it to the directory `/var/ct/cfg/`. For more information, see the **RSCT: Administration Guide**.

For information about how access controls are implemented for the IBM.LPCommands resource class and its resources, see the **lpacl** information file.

**Implementation Specifics**

This file is part of the Reliable Scalable Cluster Technology (RSCT) filesset for AIX.

**Location**

/var/ct/cfg/ctrmc.acls Contains the ctrmc.acls file

**Files**

/usr/sbin/rsct/cfg/ctrmc.acls Default location of the ctrmc.acls file

/var/ct/IW/log/mc/default Location of any errors found in the modified ctrmc.acls file
ctsec_map.global File

Purpose
Associates operating system user identifiers on the local system with network security identifiers for authorization purposes.

Description
RSCT trusted services use the identity mapping definition files ctsec_map.global and ctsec_map.local to determine whether an RSCT client application’s user should be granted access to specific RSCT functions and resources. The file is used to associate security network identifiers that are used by RSCT’s cluster security services with user identifiers on the local system. RSCT trusted services use these files to determine what association, if any, exists for the RSCT client, and then use this association while examining the RSCT access controls to determine whether the RSCT client should be granted access.

Two identity mapping definition files can be used:
- The ctsec_map.global file contains associations that are to be recognized on all nodes within the cluster configuration
- The ctsec_map.local file contains associations that are specific to a particular node

In a cluster configuration, all ctsec_map.global files should be the same. Any local system additions that are required for that specific system should be made in the ctsec_map.local file.

RSCT provides a default ctsec_map.global file in the /usr/sbin/rsct/cfg directory. Do not change this file. If you need to add more associations for the cluster, copy this file to the /var/ct/cfg directory. Make any changes to this new file: /var/ct/cfg/ctsec_map.global. Any entries that exist in the default ctsec_map.global file must exist in the replacement version of the file in the /var/ct/cfg directory, or the RSCT trusted services may refuse access to other RSCT trusted services peers. RSCT does not provide a default ctsec_map.local file. The administrator can create this file, which must reside in the /var/ct/cfg directory as well.

ctsec_map.global and ctsec_map.local are ASCII-formatted files that can be viewed and modified using a text editor. Each line in the file constitutes an entry. Blank lines and lines that start with a pound sign (#) are ignored. Each entry is used to either associate a security network identifier with a local operating system user identifier, or to expressly state that no association is allowed for a security network identifier.

Ordering of entries within these files is important. Cluster security services parses the ctsec_map.global and ctsec_map.local files as follows:
1. If the /var/ct/cfg/ctsec_map.local file exists, cluster security services checks for associations in this file
2. If the /var/ct/cfg/ctsec_map.global file exists, cluster security services checks for associations in this file
3. Otherwise, cluster security services checks for associations within the /usr/sbin/rsct/cfg/ctsec_map.global, if this file exists

The first entry that successfully grants or denies an association for a security network identifier in this search path is the one that cluster security services uses. If entries in both the ctsec_map.global and ctsec_map.local files grant differing associations to the same security network identifier, cluster security services will use the association stated by the entry in the ctsec_map.local file. Also, if two entries within the ctsec_map.global file grant different associations to the same security network identifier, cluster
security services will use the association granted by the entry listed earlier in the `ctsec_map.global` file. You can use the `ctsidmck` command to verify the association rule that is used by cluster security services for specific security network identifiers.

Cluster security services recognizes these characters as reserved: `<`, `>`, `=`, `!`, `@`, `*`, and considers these, along with white space characters, as token separators. The wildcard character `*` is permitted, but should not be used multiple times between other token separator characters. Contents of the identity mapping definition files use the following Backus-Naur format:

```
<mapping_entry> ::= <mechanism_mnemonic> ':' <mapping>
<mapping> ::= <explicit_mapping> | <mapping_rule>
<explicit_mapping> ::= <source_mapping> '=' <local_user_identity> | '!' <source_mapping>
<target_mapping> ::= <mapped_identity> | '*'
<network_identity> ::= <user_name>'@'<registry_name>
<m电站_name> ::= <user_name>'=' <local_user_identity> | '!' <local_user_identity>
<map_set> ::= <match_pattern> | '*' | <mpm_defined_reserved_word>
<match_pattern> ::= null string | <alphanumeric_string>
<alphanumeric_string> ::= any non-empty array of alphanumeric characters not consisting of the reserved token separator characters
```

An `<mpm_defined_reserved_word>` is a special instruction to the underlying security mechanism associated with the security network identifier that instructs the mechanism to interpret the identifier in a specific manner. The following reserved words are defined:

- `<iw>` A reserved word for security network identities using the RSCT host-based authentication (HBA) security mechanism. This keyword maps the HBA root network identity of the local node to the root user. When the cluster security services identity mapping program processes the `ctsec_map.global` file, it replaces the `<iw>` keyword with the node ID of the node.

- `<cluster>` A reserved word for security network identities using the HBA security mechanism. The mapping entry is applied to a security network identifier if the identifier is known to originate from any host within the cluster that is currently active for the local node.

- `<any_cluster>` A reserved word for security network identities using the HBA security mechanism. The mapping entry is applied to a security network identifier if the identifier is known to originate from any host within any cluster that the local node is currently defined. The local node does not need to be active within that cluster when the mapping is applied.

- `<realm>` A reserved word for security network identities using the Kerberos version 5 mechanism. The mapping entry is applied to a security network identity if the identifier is known to originate within the Kerberos realm that is currently active. See Restrictions.
Security

- The default identity mapping definition file /usr/sbin/rsct/cfg/ctsec_map.global is readable by all system users, but permissions prevent this file from being modified by any system user.
- When creating the override identity mapping definition files /var/ct/cfg/ctsec_map.global and /var/ct/cfg/ctsec_map.local, make sure that the files can be read by any system user, but that they can only be modified by the root user or other restrictive user identity not granted to normal system users.
- By default, these files reside in locally-mounted file systems. While it is possible to mount the /var/ct/cfg directory on a networked file system, this practice is discouraged. If the /var/ct/cfg/ctsec_map.local file were to reside in a networked file system, any node with access to that networked directory would assume that these definitions were specific to that node alone when in reality they would be shared.

Restrictions
RSCT does not support the Kerberos version 5 mechanism. Any entries using the mechanism mnemonic krb5 or the reserved word <realm> will not be applied.

Implementation Specifics
These files are part of the Reliable Scalable Cluster Technology (RSCT) cluster security services. The default file is shipped as part of the rsct.core.sec fileset for AIX.

Location
/usr/sbin/rsct/cfg/ctsec_map.global
Contains the default identity mapping definition file.

/var/ct/cfg/ctsec_map.global
Contains the replacement for the default global identity mapping definition file. Any entries that exist in the default ctsec_map.global file must be replicated in this file, or necessary access required by RSCT trusted services clients will be refused. This file contains identity mapping definitions expected to be recognized by all nodes within the cluster. It is expected that this file will have the same contents for each node within the cluster.

/var/ct/cfg/ctsec_map.local
Contains additional identity mapping definitions specific to the local node. This file adds identity mapping definitions to the set recognized for the entire cluster. Entries within this file are applied before entries from the ctsec_map.global file. It is expected that the contents of this file will vary from node to node within the cluster, and provide mappings required for clients that access the local node only.

Related Information
Commands: ctsidmck

tsec_map.local File

Purpose
Associates operating system user identifiers on the local system with network security identifiers for authorization purposes.

Description
RSCT trusted services use the identity mapping definition files ctsec_map.global and ctsec_map.local to determine whether an RSCT client application’s user should be granted access to specific RSCT functions and resources. The file is used to associate security network identifiers that are used by RSCT’s cluster security services with user identifiers on the local system. RSCT trusted services use these files to
determine what association, if any, exists for the RSCT client, and then use this association while examining the RSCT access controls to determine whether the RSCT client should be granted access.

Two identity mapping definition files can be used:

- The cttsec_map.global file contains associations that are to be recognized on all nodes within the cluster configuration
- The cttsec_map.local file contains associations that are specific to a particular node

In a cluster configuration, all cttsec_map.global files should be the same. Any local system additions that are required for that specific system should be made in the cttsec_map.local file.

RSCT provides a default cttsec_map.global file in the /usr/sbin/rsct/cfg directory. Do not change this file. If you need to add more associations for the cluster, copy this file to the /var/ct/cfg directory. Make any changes to this new file: /var/ct/cfg/ctsec_map.global. Any entries that exist in the default cttsec_map.global file must exist in the replacement version of the file in the /var/ct/cfg directory, or the RSCT trusted services may refuse access to other RSCT trusted services peers. RSCT does not provide a default cttsec_map.local file. The administrator can create this file, which must reside in the /var/ct/cfg directory as well.

ctsec_map.global and cttsec_map.local are ASCII-formatted files that can be viewed and modified using a text editor. Each line in the file constitutes an entry. Blank lines and lines that start with a pound sign (#) are ignored. Each entry is used to either associate a security network identifier with a local operating system user identifier, or to expressly state that no association is allowed for a security network identifier.

Ordering of entries within these files is important. Cluster security services parses the cttsec_map.global and cttsec_map.local files as follows:

1. If the /var/ct/cfg/ctsec_map.local file exists, cluster security services checks for associations in this file
2. If the /var/ct/cfg/ctsec_map.global file exists, cluster security services checks for associations in this file
3. Otherwise, cluster security services checks for associations within the /usr/sbin/rsct/cfg/ctsec_map.global, if this file exists

The first entry that successfully grants or denies an association for a security network identifier in this search path is the one that cluster security services uses. If entries in both the cttsec_map.global and cttsec_map.local files grant differing associations to the same security network identifier, cluster security services will use the association stated by the entry in the cttsec_map.local file. Also, if two entries within the cttsec_map.global file grant different associations to the same security network identifier, cluster security services will use the association granted by the entry listed earlier in the cttsec_map.global file. You can use the ctsidmck command to verify the association rule that is used by cluster security services for specific security network identifiers.

Cluster security services recognizes these characters as reserved: <, >, :, =, !, @, *, and considers these, along with white space characters, as token separators. The wildcard character * is permitted, but should not be used multiple times between other token separator characters. Contents of the identity mapping definition files use the following Backus-Naur format:

```
<mapping_entry>::= <mechanism_mnemonic> ':' <mapping>

<mechanism_mnemonic>::= 'unix', 'krb5'

<mapping>::= <explicit_mapping> | <mapping_rule>

<explicit_mapping>::= <source_mapping> '=' <local_user_identity>
| '!=' <source_mapping>

<source_mapping>::= <network_identity> | <match_pattern>
```
An `<mpm_defined_reserved_word>` is a special instruction to the underlying security mechanism associated with the security network identifier that instructs the mechanism to interpret the identifier in a specific manner. The following reserved words are defined:

- `<iw>`: A reserved word for security network identities using the RSCT host-based authentication (HBA) security mechanism. This keyword maps the HBA root network identity of the local node to the root user. When the cluster security services identity mapping program processes the `ctsec_map.global` file, it replaces the `<iw>` keyword with the node ID of the node.

- `<cluster>`: A reserved word for security network identities using the HBA security mechanism. The mapping entry is applied to a security network identifier if the identifier is known to originate from any host within the cluster that is currently active for the local node.

- `<any_cluster>`: A reserved word for security network identities using the HBA security mechanism. The mapping entry is applied to a security network identifier if the identifier is known to originate from any host within any cluster that the local node is currently defined. The local node does not need to be active within that cluster when the mapping is applied.

- `<realm>`: A reserved word for security network identities using the Kerberos version 5 mechanism. The mapping entry is applied to a security network identifier if the identifier is known to originate within the Kerberos realm that is currently active. See Restrictions.

**Security**

- The default identity mapping definition file `/usr/sbin/rsct/cfg/ctsec_map.global` is readable by all system users, but permissions prevent this file from being modified by any system user.
- When creating the override identity mapping definition files `/var/ct/cfg/ctsec_map.global` and `/var/ct/cfg/ctsec_map.local`, make sure that the files can be read by any system user, but that they can only be modified by the root user or other restrictive user identity not granted to normal system users.
- By default, these files reside in locally-mounted file systems. While it is possible to mount the `/var/ct/cfg` directory on a networked file system, this practice is discouraged. If the `/var/ct/cfg/ctsec_map.local` file were to reside in a networked file system, any node with access to that networked directory would assume that these definitions were specific to that node alone when in reality they would be shared.

**Restrictions**

RSCT does not support the Kerberos version 5 mechanism. Any entries using the mechanism mnemonic `krb5` or the reserved word `<realm>` will not be applied.
Implementation Specifics

These files are part of the Reliable Scalable Cluster Technology (RSCT) cluster security services. The default file is shipped as part of the rsct.core.sec fileset for AIX.

Location

/usr/sbin/rsct/cfg/ctsec_map.global
Contains the default identity mapping definition file.

/var/ct/cfg/ctsec_map.global
Contains the replacement for the default global identity mapping definition file. Any entries that exist in the default ctsec_map.global file must be replicated in this file, or necessary access required by RSCT trusted services clients will be refused. This file contains identity mapping definitions expected to be recognized by all nodes within the cluster. It is expected that this file will have the same contents for each node within the cluster.

/var/ct/cfg/ctsec_map.local
Contains additional identity mapping definitions specific to the local node. This file adds identity mapping definitions to the set recognized for the entire cluster. Entries within this file are applied before entries from the ctsec_map.global file. It is expected that the contents of this file will vary from node to node within the cluster, and provide mappings required for clients that access the local node only.

Related Information

Commands: ctsidmck

Data (D.*) Files for BNU

Purpose

Contain data to be sent to remote systems.

Description

Data (D.*) files contain the data to be sent to remote systems by the Basic Networking Utilities (BNU) uucico daemon. The full path name of a data file is a form of the following:

/var/spool/uucp/SystemName/D.SystemNamexxxx###

where the SystemName directory and the SystemName portion of the file name indicate the name of the remote system. The xxxxx### notation is the hexadecimal sequence number of the command (C.*) file associated with that data file, for example: D.venus471af8.

After a set period of time (specified by the uusched daemon), the uucico daemon transfers the data file to the designated system. It places the original data file in a subdirectory of the BNU spooling directory named /var/spool/uucp/SystemName where the SystemName directory is named for the computer that is transmitting the file, and creates a temporary (TM.*) file to hold the original data file.

After receiving the entire file, the BNU program takes one of the three following actions:

• If the file was sent with the uucp command and there were no transfer problems, the program immediately renames the TM.* file with the appropriate data file name, such as D.venus471af8, and sends it to the specified destination.

• If the file was sent with the uuto command, the BNU program also renames the temporary data file with the appropriate D.* file name. The program then places the data file in the /var/spool/uucppublic public directory, where the user receives the data file and handles it with one of the uupick command options.
• If there were transfer problems (such as a failed login or an unavailable device), the temporary data file remains in the spooling subdirectory. The `uudemon.cleanu` command, a shell procedure, removes these files automatically at specified intervals. They can also be removed manually.

**Files**

<table>
<thead>
<tr>
<th>Directory Path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/etc/uucp/Systems</td>
<td>Describes accessible remote systems.</td>
</tr>
<tr>
<td>/var/spool/uucp/SystemName</td>
<td>Contains BNU command, data, and execute files.</td>
</tr>
<tr>
<td>/var/spool/uucp/SystemName/C.*</td>
<td>Contains instructions for file transfers.</td>
</tr>
<tr>
<td>/var/spool/uucp/SystemName/TM.*</td>
<td>Stores data files temporarily after they have been transferred to a remote system.</td>
</tr>
<tr>
<td>/var/spool/uucppublic/*</td>
<td>Contains files that the BNU program has transferred.</td>
</tr>
</tbody>
</table>

**Related Information**
The `uucp` command, `uudemon.cleanu` command, `uupick` command, `uuto` command, `uux` command.
The `uucico` daemon, `uusched` daemon, `uuxqt` daemon.

**/dev/hty File**

**Purpose**
Defines the Network Terminal Accelerator adapter tty interface.

**Description**
The `/dev/hty` device files define, for the host computer, the interface-to-host adapter communication channels. For each I/O device connected to the host computer through a host adapter, there must be a `/dev/hty` device file created to allow communication between the host computer and the I/O device.

To allow for future expansion, there may be more `/dev/hty` files than actual physical devices connected through the host adapter.

The hty ports are functionally equivalent to `/dev/tty` device files. The minor number corresponds to the channel number, as defined in the `hty_config` file.

**Files**

<table>
<thead>
<tr>
<th>File Path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/dev/hty</td>
<td>Specifies the path to the file.</td>
</tr>
<tr>
<td>/dev/rhp*</td>
<td>Adapter raw device.</td>
</tr>
</tbody>
</table>

**Related Information**
The `/dev/rhp` file.

**/dev/rhp File**

**Purpose**
Defines the Network Terminal Accelerator adapter raw interface.
Description
The /dev/rhp* device files define, for the host computer, the interface to the host adapters. For each host adapter installed in the host computer, there must be a /dev/rhp* device file created in order to allow communication between the host computer and the host adapter board.

The /dev/rhp* device file corresponding to a respective host adapter is used as an argument in many of the utility programs.

Files
/dev/rhp       Specifies the path to the file
/dev/hty       Defines the Network Terminal Accelerator adapter tty interface.

Related Information
The /dev/hty file.

DHCP Client Configuration File

Purpose
Default configuration information for the Dynamic Host Configuration Protocol (DHCP) client program (dhcpcd).

Description
The dhcpcd configuration file contains entries for logging information, requested options, interfaces to configure, and other items.

Following are the formats for the data in the configuration file.

# Comment line
   The # character means that there is a comment from that point to the end of the line.

numLogFiles n
   Specifies the number of log files. If 0 is specified, no log file will be maintained and no log message is displayed anywhere. n is the maximum number of log files maintained as the size of the most recent log file reaches its maximum size and a new log file is created.

logFileSize n
   Maximum size of a log file. When the size of the most recent log file reaches this value, it is renamed and a new log file is created. n is measured in kilobytes(KB).

logFileName filename
   Name and path of the most recent log file. Less recent log files have the number 1 to (n - 1) appended to their names; the larger the number, the older the file.

logItem <option name>
   One item that will be logged. Multiple of these lines are allowed. This allows for the specified logging level to be turned on. The following are option names:

   SYSERR
      System error, at the interface to the platform

   OBJERR
      Object error, in between objects in the process

   PROTERR
      Protocol error, between client and server
interface <ifName>
The interface to configure DHCP on. This may be the interface that is to be configured. Multiples of these are allowed. There is a special entry, any. This tells the DHCP client to configure the first one it finds and completes successfully. If the any option is used, there should not be any other interface specified. The interface statement may be immediately followed by a pair of curly braces, in which the options requested for this interface can be specified. Options requested within interface curly braces apply only to this interface. See [DHCP Server Configuration File] for a list of options and formats.

clientid <MAC | HOSTNAME>
Specifies the client id to use in all communication with the server. MAC denotes that the hardware address for the particular interface should be used as the client id. HOSTNAME denotes that the domain host name should be used as the client id. The default is MAC.

sniffer <exec string>
Specifies a string enclosed in quotes, indicating a program to execute to detect hardware failure/recovery for an interface. The dhcp client will look for signal 23(SIGIO) to indicate that the network interface is up and signal 16(SIGURG) to indicate that the network interface is down.

option <code> [value] [exec <string>]
Specifies an option requested by this client. Its scope is determined by whether it is inside a set of curly braces for a particular interface, or if it is outside all curly braces. If outside, it applies to all interfaces. code is the option code of the option requested. value is the requested value for that option. This value is passed to the server with the option. The value is not required. The keyword exec denotes a string following which should be executed if this option is returned by the server. This string is expected to be an executable shell script or program. An "%s" may be included in the string. If present, the value returned by the server will be provided in ascii.

vendor Specifies the special syntax for the specification of the vendor extensions field. It is followed by a set of curly braces. Inside the curly braces, the options and values for the vendor extensions field are specified. The exec string on an option inside the vendor extensions options is not valid. It is ignored.

reject <code>
Specifies that if this option code is returned by the server, this option should be ignored by the client. Its value should not be used.

otherOptions <accept | reject>
Specifies how all other options should be handled by the client. This refers to any options not specifically requested with an "option" statement or rejected with a "reject" statement. The default is that all options are accepted.

updateDNS <string>
A string enclosed in quotes, indicating a program to execute to update the DNS server with the
new inverse mapping for the IP address and names served by `dhcp`. This string should include four `%s`'s to indicate the placement of the following information from the `dhcp` client:

**hostname**
Value of option 12. The value returned by the `dhcp` server is used, if one is supplied. Else, if the client specified a value in *this* file, the client-requested value is used. If neither the client specified a requested hostname nor the server supplied one, this exec string will not be executed.

**domainname**
Value of option 15. The value returned by the `dhcp` server is used, if one is supplied. Else, if the client specified a value in *this* file, the client-requested value is used. If neither the client specified a requested hostname nor the server supplied one, a null string (""") will be supplied by `dhcp`. Therefore, this value is optional.

**Ip Address**
IP address leased to this client by the server. The string is supplied in dotted notation, for example, 9.2.23.43.

**leasetime**
Lease time granted by the server. This string is a decimal number representing the number of seconds of the lease.

These values are output by `dhcp` in this order:

```
hostname domainname Ip Address leasetime
```

A script `/usr/sbin/dhcpaction` has been provided with this function, as well as actions to help NIM interact with DHCP clients. Run the script as follows:

```
/usr/sbin/dhcpaction hostname domainname ipaddress
leasetime < A | PTR | BOTH | NONE > NONIM
```

The first four parameters are what will be used to update the DNS server. The fifth parameter tells `dhcpaction` to update the A record, the PTR record, or both, or none. The options are A, PTR, BOTH, NONE. The sixth parameter is used to tell servers that NIM is being used, and processing needs to be done when a client changes address. The options for this are NIM and NONIM. On clients, this must be set to NONIM.

An example follows:

```
updateDNS "#/usr/sbin/dhcpaction %s %s %s %s %s PTR
NONIM 2>&1 >>/tmp/updns.out"
```

`initTimeout <timeout>`
Specifies the timeout value in minutes. If the dhcp client fails to configure an address for an interface within this timeout value, it stops making further attempts. This entry applies to systems running AIX 5.2 and subsequent releases.

This file is part of TCP/IP in Network Support Facilities in Base Operating System (BOS) Runtime.

**Example**
This example tells the `dhcpcd` daemon to use log files of a maximum of 100Kb in size and at most four of them.

The base name for the log files is `/usr/tmp/dhcpsd.log`. The user also would like to only log four of the nine possible log entry types. The user also specified a string to use for updating the Dynamic Domain Name Server. The user also specified that the `clientid` to the server should be based on the mac-address of the interface adapter that is trying to be configured. The user also specified that all options should be accepted and instantiated (otheroptions accept), except for option 9 (reject 9).
The options the user specified were the domain (option 15), but since this option is global to the interface keywords, it applies to both interfaces.

Inside each interface, the hostname is specified with option 12.

```
numLogFiles 4
logFileSize 100
logFileName /usr/tmp/dhcpsd.log
logItem SYSERR
logItem OBJERR
logItem PROTERR
logItem TRACE
updateDNS "nsupdate -h%s -d%s -i% -s"
clientid MAC
otheroptions accept
reject 9
option 15 "austin.ibm.com"
interface en0
{
    option 12 "e-chisos"
}
interface tr0
{
    option 12 "t-chisos"
}
```

Related Information

The `dhcpcd` Daemon

The **DHCP Server Configuration File**

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

TCP/IP problems with Dynamic Host Configuration Protocol in *Networks and communication management.*

**DHCP Server Configuration File**

**Purpose**

Defines default configuration information for the Dynamic Host Configuration Protocol (DHCP) server program (dhcpsd).

**Description**

The dhcpsd configuration file contains entries for logging information, options to return, machines to configure, and other items.

Following are the formats for the data in the configuration file.

**# Comment line**

The # character means that there is a comment from that point to the end of the line.

```
## "Name of Resource" "<Keyword> <value> <value> ..."
```

The ## characters denote a named resource. This is used by the dhcpsconf program to allow the user to create specific resources. The data is stored in the server file so that it can be read in with the configuration file and displayed as the name and not the value in the viewing window of dhcpsconf.
The format of the ## line is a quoted string that is the name of the resource followed by a double-quoted string representing a valid possible line for a configuration file. The second quoted string should be syntactically correct for a line in a DHCP server configuration file. The keyword can only be option, network, subnet, class, and client.

### "DHCP Server" "Any line from a server file"

The ### characters denote a server configuration file. This allows for multiple server files to be saved in one file. The dhcpsconf program uses this to present multiple server datasets in a master. This would be useful, if you were to define a network with 10 servers and wanted to save all the server information in one file and maintain a default server. The default server would go into the master file, and the servers would be saved in the master file with the ### characters. The dhcpsconf program has a function that allows you to create a specific server configuration out of the master file.

numLogFiles n
Specifies the number of log files. If 0 is specified, no log file will be maintained and no log message is displayed anywhere. n is the maximum number of log files maintained as the size of the most recent log file reaches its maximum size and a new log file is created.

logFileSize n
Maximum size of a log file. When the size of the most recent log file reaches this value, it is renamed and a new log file is created. n is measured in kilobytes(KB).

logFileName filename
Name and path of the most recent log file. Less recent log files have the number 1 to (n - 1) appended to their names; the larger the number, the older the file.

logItem <option name>
One item that will be logged. Multiple of these lines are allowed. This allows for the specified logging level to be turned on. The following are option names:

- **SYSERR**
  System error, at the interface to the platform
- **OBJERR**
  Object error, in between objects in the process
- **PROTERR**
  Protocol error, between client and server
- **WARNING**
  Warning, worth attention from the user
- **EVENT**
  Event occurred to the process
- **ACTION**
  Action taken by the process
- **INFO**
  Information that might be useful
- **ACNTING**
  Who was served, and when
- **TRACE**
  Code flow, for debugging.

clientrecorddb <filename>
This is the path to a file to substitute for /etc/dhcps.cr. Configurations that support a large number of addresses should set clientrecorddb and addressrecorddb database files in a file system with substantial free space.
addressrecorddb <filename>
This is the path to a file to substitute for /etc/dhcps.ar.

network <Network address> [<Subnet Mask>|<range>]
Specifies one network administered by this server. Network address is the address of this network.
This address is specified in the dotted notation (for example, 9.0.0.0, 128.81.0.0, or 192.81.20.0).
Full four-byte value should be specified (for example, 9, 128.81, or 192.81.20 is not legal).

Network address may optionally be followed by the subnet mask, a range, or nothing.

If a subnet mask is specified, one or more subnet statements should appear in the succeeding
lines within a pair of curly braces. The subnet mask may be specified either in the dotted notation
(for example, 255.255.255.128) or as a number indicating the number of 1 bits in the mask (for
example, 25, which is equivalent to 255.255.255.128). The means that a network is not a
collection of all subnet for a network, but all subnets with the same length subnet for that network
“prefix.”

If a range is specified, it determines, within the network, the range of hosts that are administered
by this server, and it implies that there is no subnetting. A range is specified by the host
addresses, in the dotted notation, at the lower end and the higher end of the range, respectively,
separated by a hyphen with no spaces before or after it (for example, 192.81.20.1-129.81.20.128).
A range must encompass all addresses to be administered because multiple network statements
to define the same network are not allowed. Use the “client” statement to exclude any addresses
in the range that the server should not administer.

If nothing is specified after Network address, all hosts in that network are administered by this
server.

A network statement may be immediately followed by a pair of curly braces, in which parameters
(for example, options) particular to this network can be specified.

subnet <Subnet address> [<range>]
One or more subnet statements are enclosed by a pair of curly braces that immediately follows a
network statement with subnet mask. A subnet statement specifies one subnet within that network.

Subnet address is the address of this subnet. This address is specified in the dotted notation (for
example, 9.17.32.0 or 128.81.22.0).

Subnet address may be followed by a range or nothing.

If a range is specified, it determines, within the subnet, the range of hosts that are administered by
this server. A range is specified by the host addresses, in the dotted notation, at the lower end and
the higher end of the range, respectively, separated by a hyphen with no spaces before or after it.
A range must encompass all addresses to be administered since multiple subnet statements to
define the same subnet are not allowed. Use the “client” statement to exclude any addresses in
the range which the server should not administer.

If nothing is specified after Subnet address, all hosts in that subnet are administered by this
server.

The ranges in two servers administering the same subnet cannot overlap. Otherwise, two hosts
may be assigned the same address.

A subnet statement may be immediately followed by a pair of curly braces, in which parameters
(for example, options) particular to this subnet can be specified.

class <class_name> [<range>]
Specifies a class. The class name is a simple ascii string. A class’s scope is determined by the
curly braces in which it is enclosed. If it is outside all curly braces, then its scope is the entire file.

A class name may be followed by a range or nothing. If a range of Ip Addresses is specified, then
only addresses in that range will be assigned to clients who request this class. Note that clients
who request this class, for which the subnet does not match the range, will not be processed. Bad
addresses will not be given out by the server. If an address range is not specified, then addresses
will be given to clients using the usual rules of assignment (by network clauses).

The class statement may be immediately followed by a pair of curly braces, in which the options
particular to this class can be specified. A class may be defined within the curly braces of a
subnet, but a subnet may not be defined within the curly braces of a class.

Options set up in the network or subnet containing a class definition will also apply to the class.

```
client <id_type> <id_value> <address>
```

Specifies a definition of client/address processing.

<id_type> is 0 for a string, otherwise it is one of the hardware types defined in RFC 1340 (for
example, 6 for IEEE 802 networks.)

<id_value> is a character string for <id_type>=0. Typically, this would be a domain name. For a
non-zero <id_type>, the <id_value> is a hexadecimal string representing the hardware address of
the client.

**Note:** An <id_type> of 0 and an <id_value> of 0 indicates that the <address> specified should not
be distributed by this server.

The <address> can be the string "none" to indicate that the client with <id_type> and <id_value>
should not be serviced by this server. The <address> can be the string "any" to indicate that the
server should choose an appropriate address for this client. The <address> can be an internet
address in dotted notation (for example, 9.2.15.82). This will be the Ip address given to the
particular client specified by <id_type> and <id_value>. As mentioned above, an <id_type> of 0
and an <id_value> of 0 indicates that the <address> specified should not be distributed by this
server.

**Note:** If a client is configured in this way on the server, then any class information requested by
the client will be ignored. No class-specific information will be processed for these clients.

The client statement may be immediately followed by a pair of curly braces, in which the options
particular to this client can be specified.

A client statement with an address specified that is not part of the address pool specified in a
network/subnet elsewhere in this file must contain the subnet mask option(1). For all other clients,
the server will compute the subnet mask option to send the client based on the network/subnet
definitions.

**Note:** All clients inherit all globally defined options. A client defined in a network scope will inherit
options defined for that network. A client defined in a subnet scope, will inherit options
defined for that subnet and encompassing network.

A class definition inside a client scope is not allowed.

The client statement may be used to configure **bootp** clients. To do this, specify all the **bootp**
options using the option syntax defined below. In addition, specify an infinite lease time in the
client scope with "option 51 0xffffffff". DHCP options will not be served to the **bootp** client.

```
nakoutofrange <option>
```

Specifies the behavior of the DHCP server when a client requests an IP address this server does
not administer.

When a DHCP client moves from one DHCP domain to a new DHCP domain, the client requests
its previous IP address from the new DHCP server. By default, the DHCP server ignores such
requests without sending a refusal (DHCP NAK). This default behavior is chosen because there
remains a remote possibility that there might be other DHCP servers on the network, which might
be able to service that request. But many common DHCP client implementations are such that when a DHCP REQUEST message is unanswered, the client continues using its previous address and network configuration, which is most likely useless in the new environment. The nakoutofrange policy in the AIX DHCP server enables the DHCP administrator to force the server to send the refusals to clients in such situations, and to force the clients to request new addresses and configurations.

The nakoutofrange option can be placed either inside a subnet scope or outside, in the global scope. Each option and its placement have specific implications to the behavior of the DHCP server. The following are the options available.

**none**
This is the default setting even if the nakoutofrange keyword is not present. A DHCP server only sends a DHCP NAK response when a client asks for an address that the server controls. However, for some reasons the client is not allowed to use the address. For example, a client requests an address that another client is using.

**insubnet**
The DHCP server sends a DHCP NAK response to a client that is requesting an address that matches the subnet from which the client is broadcasting. However, the address does not fall within the controlled IP address ranges of that subnet.

**Note:** Do not use this option when multiple DHCP servers are controlling separate IP address ranges on the same subnet, unless every access point to the subnet can only reach one of the DHCP servers.

**notinsubnet**
The DHCP server sends a DHCP NAK response to a client that is requesting an address that does not match the subnet from which the client is broadcasting. When this option is placed outside the subnet scope, the DHCP NAK response is sent when the client’s broadcast is received from a subnet that is not controlled by the server.

**Note:** When placed inside the subnet scope, this option should be used when there are no other logical subnets on the client’s network segment that are administered through any DHCP server.

**both**
The DHCP server sends a DHCP NAK response to a client that is requesting an address that the server is unable to assign to the client, regardless of the reason.

**Note:** Placing this option inside a subnet scope combines the implications of placing the notinsubnet and insubnet options inside the subnet scope. Place this option inside a subnet scope when the configured DHCP server is the only DHCP server capable of assigning an address to a client broadcasting from any of the network access points on the subnet. This option should be placed globally only when the configured DHCP server is the only reachable DHCP server from any of the network access points.

The following is an example of use of nakoutofrange option.

```
nakoutofrange none
    subnet 192.1.1.0 255.255.255.0 192.1.1.5-192.1.1.254
    {
        nakoutofrange both
            # ... options ...
    }
    subnet 12.1.10.0 255.255.255.0 12.1.10.5-12.1.10.150
    {
        # another server controls addresses 12.1.10.151-12.1.10.254
        nakoutofrange notinsubnet
            # ... options ...
    }
```
**Note:** A typical configuration is to use the both option inside all subnet scopes and to leave the default option none enforced in the global scope. Using a policy other than none in the global area requires a firm understanding of the network topology and the ability that DHCP server can be reached from any network access point by clients of the DHCP server.

```plaintext
option <code> <value>
```

This parameter specifies the value of an option defined in “DHCP Options and BOOTP Vendor Extensions” (RFC 1533) and supported by this server.

An option is specified by the “option” keyword followed by the option code of this option and its data field, in a single line. One or more of this parameter may be specified.

The scope within which an option applies is delimited by a pair of curly braces ({, }) surrounding this parameter.

Two or more options with the same option code may be specified. Their data fields are concatenated in a single option in a packet generated by the server if the options have the same scope or one’s scope includes that of another.

Some of the defined options do not need to be specified by this parameter. These options are either mandated by the protocol or this implementation to be present in proper packets, or only generated by a client. These options are:

<table>
<thead>
<tr>
<th>Option Code</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Pad Option</td>
</tr>
<tr>
<td>255</td>
<td>End Option</td>
</tr>
<tr>
<td>1</td>
<td>Subnet Mask</td>
</tr>
<tr>
<td>50</td>
<td>Request IP Address</td>
</tr>
<tr>
<td>51</td>
<td>IP Address Lease Time</td>
</tr>
<tr>
<td>52</td>
<td>Option Overload</td>
</tr>
<tr>
<td>53</td>
<td>DHCP Message Type</td>
</tr>
<tr>
<td>54</td>
<td>Server Identifier</td>
</tr>
<tr>
<td>55</td>
<td>Parameter Request List</td>
</tr>
<tr>
<td>57</td>
<td>Maximum DHCP Message Size</td>
</tr>
<tr>
<td>58</td>
<td>Renewal (T1) Time Value</td>
</tr>
<tr>
<td>59</td>
<td>Rebinding (T2) Time Value</td>
</tr>
<tr>
<td>60</td>
<td>Class identifier of client</td>
</tr>
<tr>
<td>61</td>
<td>Client identifier</td>
</tr>
</tbody>
</table>

The other options may be specified by this parameter.

When specifying an option, its data field takes one of the following formats:

**IP Address**

```
xxx.xxx.xxx.xxx
```

**IP Addresses**

```
[xxx.xxx.xxx.xxx ...
```

**IP Address Pair**

```
[ip address:ip address]
```

**IP Address Pairs**

```
[[ip address:ip address] ...]
```

Chapter 1. System Files 51
Boolean
  [0, 1]

Byte  [-128, 127]

Unsigned Byte
  [0, 255]

Unsigned Bytes
  [[0, 255] [0, 255] ...]

Short  [-32768, 32767]

Unsigned Short
  [0, 65535]

Unsigned Shorts
  [[0, 65535] [0, 65536]

Long  [-2147483648, 2147483647]

Unsigned Long
  [0, 4294967295]

String  "Value Here"

Note: All IP addresses are specified in dotted-decimal form.

Each of the defined options is listed below by its code and name, followed by the format of its data field. These are specified in latest Vendor Extensions RFC.

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Data Field Format and Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Pad Option</td>
<td>No need to specify</td>
</tr>
<tr>
<td>255</td>
<td>End Option</td>
<td>No need to specify</td>
</tr>
<tr>
<td>1</td>
<td>Subnet Mask</td>
<td>Unsigned Long</td>
</tr>
<tr>
<td>2</td>
<td>Time Offset</td>
<td>Long</td>
</tr>
<tr>
<td>3</td>
<td>Router Option</td>
<td>IP Addresses</td>
</tr>
<tr>
<td>4</td>
<td>Timer Server Option</td>
<td>IP Addresses</td>
</tr>
<tr>
<td>5</td>
<td>Name Server Option</td>
<td>IP Addresses</td>
</tr>
<tr>
<td>6</td>
<td>Domain Name Server Option</td>
<td>IP Addresses</td>
</tr>
<tr>
<td>7</td>
<td>Log Server Option</td>
<td>IP Addresses</td>
</tr>
<tr>
<td>8</td>
<td>Cookie Server Option</td>
<td>IP Addresses</td>
</tr>
<tr>
<td>9</td>
<td>LPR Server Option</td>
<td>IP Addresses</td>
</tr>
<tr>
<td>10</td>
<td>Impress Server Option</td>
<td>IP Addresses</td>
</tr>
<tr>
<td>11</td>
<td>Resource Location Server Option</td>
<td>IP Addresses</td>
</tr>
<tr>
<td>12</td>
<td>Host Name Option</td>
<td>String</td>
</tr>
<tr>
<td>13</td>
<td>Boot File Size Option</td>
<td>Unsigned Short</td>
</tr>
<tr>
<td>14</td>
<td>Merit Dump File</td>
<td>String</td>
</tr>
<tr>
<td>15</td>
<td>Domain Name</td>
<td>String</td>
</tr>
<tr>
<td>16</td>
<td>Swap Server</td>
<td>IP Address</td>
</tr>
<tr>
<td>17</td>
<td>Root Path</td>
<td>String</td>
</tr>
<tr>
<td>18</td>
<td>Extensions Path</td>
<td>String</td>
</tr>
</tbody>
</table>
# IP Layer Parameters per Host

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Data Field Format and Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>19</td>
<td>IP Forwarding Enable/Disable Option</td>
<td>Boolean</td>
</tr>
<tr>
<td>20</td>
<td>Non-local Source Routing Enable/Disable Option</td>
<td>Boolean</td>
</tr>
<tr>
<td>21</td>
<td>Policy Filter Option</td>
<td>IP Address Pairs</td>
</tr>
<tr>
<td>22</td>
<td>Maximum Datagram Reassembly Size</td>
<td>Unsigned Short</td>
</tr>
<tr>
<td>23</td>
<td>Default IP Time-to-live</td>
<td>Unsigned Byte</td>
</tr>
<tr>
<td>24</td>
<td>Path MTU Aging Timeout Option</td>
<td>Unsigned Long</td>
</tr>
<tr>
<td>25</td>
<td>Path MTU Plateau Table</td>
<td>Unsigned Shorts</td>
</tr>
</tbody>
</table>

# IP Layer Parameters per Interface

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Data Field Format and Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>26</td>
<td>Interface MTU Option</td>
<td>Unsigned Short</td>
</tr>
<tr>
<td>27</td>
<td>All Subnets are Local Option</td>
<td>Boolean</td>
</tr>
<tr>
<td>28</td>
<td>Broadcast Address Option</td>
<td>IP Address</td>
</tr>
<tr>
<td>29</td>
<td>Perform Mask Discovery Option</td>
<td>Boolean</td>
</tr>
<tr>
<td>30</td>
<td>Mask Supplier Option</td>
<td>Boolean</td>
</tr>
<tr>
<td>31</td>
<td>Perform Router Discovery Option</td>
<td>Boolean</td>
</tr>
<tr>
<td>32</td>
<td>Router Solicitation Address Option</td>
<td>IP Address</td>
</tr>
<tr>
<td>33</td>
<td>Static Route Option</td>
<td>IP Address Pairs</td>
</tr>
</tbody>
</table>

# Link Layer Parameters per Interface

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Data Field Format and Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>34</td>
<td>Trailer Encapsulation Option</td>
<td>Boolean</td>
</tr>
<tr>
<td>35</td>
<td>ARP Cache Timeout Option</td>
<td>Unsigned Long</td>
</tr>
<tr>
<td>36</td>
<td>Ethernet Encapsulation Option</td>
<td>Boolean</td>
</tr>
</tbody>
</table>

# TCP Parameters

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Data Field Format and Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>37</td>
<td>TCP Default TTL Option</td>
<td>Unsigned Byte</td>
</tr>
<tr>
<td>38</td>
<td>TCP Keepalive Interval Option</td>
<td>Unsigned Long</td>
</tr>
<tr>
<td>39</td>
<td>TCP Keepalive Garbage Option</td>
<td>Boolean</td>
</tr>
</tbody>
</table>

# Application and Service Parameters

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Data Field Format and Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>NIS Domain Option</td>
<td>String</td>
</tr>
<tr>
<td>41</td>
<td>NIS Option</td>
<td>IP Addresses</td>
</tr>
<tr>
<td>42</td>
<td>Network Time Protocol Servers Option</td>
<td>IP Addresses</td>
</tr>
<tr>
<td>Code</td>
<td>Name</td>
<td>Data Field Format and Notes</td>
</tr>
<tr>
<td>------</td>
<td>------------------------------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>43</td>
<td>Vendor Specific Information</td>
<td>Unsigned Bytes</td>
</tr>
<tr>
<td>44</td>
<td>NetBIOS over TCP/IP Name Server Option</td>
<td>IP Addresses</td>
</tr>
<tr>
<td>45</td>
<td>NetBIOS over TCP/IP Datagram Distribution Server</td>
<td>IP Addresses</td>
</tr>
<tr>
<td>46</td>
<td>NetBIOS over TCP/IP Node Type Option</td>
<td>Unsigned Byte</td>
</tr>
<tr>
<td>47</td>
<td>NetBIOS over TCP/IP Scope Option</td>
<td>Unsigned Bytes</td>
</tr>
<tr>
<td>48</td>
<td>X Window System Font Server Option</td>
<td>IP Addresses</td>
</tr>
<tr>
<td>49</td>
<td>X Window System Display Manager Option</td>
<td>IP Addresses</td>
</tr>
</tbody>
</table>

DHCP Extensions

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Data Field Format and Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>Request IP Address</td>
<td>No need to specify</td>
</tr>
<tr>
<td>51</td>
<td>IP Address Lease Time</td>
<td>Unsigned Long</td>
</tr>
<tr>
<td>52</td>
<td>Option Overload</td>
<td>No need to specify</td>
</tr>
<tr>
<td>53</td>
<td>DHCP Message Type</td>
<td>No need to specify</td>
</tr>
<tr>
<td>54</td>
<td>Server Identifier</td>
<td>No need to specify</td>
</tr>
<tr>
<td>55</td>
<td>Parameter Request List</td>
<td>No need to specify</td>
</tr>
<tr>
<td>56</td>
<td>Message</td>
<td>String</td>
</tr>
<tr>
<td>57</td>
<td>Maximum DHCP Message Size</td>
<td>No need to specify</td>
</tr>
<tr>
<td>58</td>
<td>Renewal (T1) Time Value</td>
<td>No need to specify</td>
</tr>
<tr>
<td>59</td>
<td>Rebinding (T2) Time Value</td>
<td>No need to specify</td>
</tr>
<tr>
<td>60</td>
<td>Class Identifier of Client</td>
<td>Generated by client</td>
</tr>
<tr>
<td>61</td>
<td>Client Identifier</td>
<td>Generated by client</td>
</tr>
</tbody>
</table>

BOOTP Specific Options

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Data Field Format and Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>sa</td>
<td>Server Address for the BOOTP client to use</td>
<td>IP Address</td>
</tr>
<tr>
<td>bf</td>
<td>Bootfile for the BOOTP client to use</td>
<td>String</td>
</tr>
<tr>
<td>hd</td>
<td>Home Directory for the BOOTP client to search</td>
<td>String</td>
</tr>
</tbody>
</table>

Following is an example of BOOTP specific options:

```
option sa 1.1.2.2
option hd "/vikings/native"
option bf "bootfile.asdg"
```

Other option numbers may be specified, up to a maximum of 255. The options not listed above must be specified with the unsigned byte list type. Following is an example:

```
option 178 01 34 53 # Means place tag 178 with value 0x013453
```
leaseTimeDefault <amount>[<unit>]
Specifies the default lease duration for the leases issued by this server. In the absence of any more specific lease duration (for example, lease duration for specific client(s) or class of clients), the lease duration specified by this parameter takes effect.

The amount is specified by a decimal number. The unit is one of the following (plural is accepted):
- year
- month
- week
- day
- hour
- minute (default if unit is absent)
- second

There is at least one white space in between the amount and unit. Only the first amount following the keyword has effect.

If this parameter is not specified, the default lease duration is one (1) hour.

This parameter should appear outside of any pair of curly braces, for example, it applies to all leases issued by this server.

Note: This keyword only applies to the default for all addresses. To specify a specific lease time for a subnet, network, class or client, use the usual "option 51 value" to specify that lease time (in seconds).

leaseExpireInterval <amount> [ <unit> ]
Specifies the time interval at which the lease expiration condition is examined, and if a running lease meets such condition, it is expired. The value of this parameter applies to all leases administered by this server.

The amount is specified by a decimal number. The unit is one of the following (plural is accepted):
- year
- month
- week
- day
- hour
- minute (default if unit is absent)
- second

There is at least one white space in between the amount and unit. Only the first amount following the keyword has effect.

If this parameter is not specified, the default interval is one (1) minute.

This parameter should appear outside of any pair of curly braces, for example, it applies to all leases issued by this server.

The value of this parameter should be in proportion with that of parameter leaseTimeDefault so that the expirations of leases are recognized in time.

supportBOOTP [yes | no]
Indicates to the server whether or not to support requests from BOOTP clients.

If yes is specified, the server will support BOOTP clients.

If the value field is not a yes, or the keyword is omitted, the server will not support BOOTP clients.

The scope of this parameter covers all the networks and subnets administered by this server.
If the server previously supported BOOTP clients and has been reconfigured not to support
BOOTP clients, the address binding for a BOOTP client established before the reconfiguration, if
any, will still be maintained until the time when that BOOTP client sends a request again (when it
is rebooting.) At that time, the server will not respond, and the binding will be removed.

**supportunlistedClients [yes | no]**

Indicates to the server whether or not to support requests from clients that are not specifically
configured with their own individual client statements in the server.

If yes is specified, the server will support unlisted clients.

If the value field is anything other than yes, the server will not support unlisted clients.

If this keyword is not found in the file, the server will support clients not specifically configured with
a client statement.

**updateDNS <string>**

A string enclosed in quotes, indicating a program to execute to update the DNS server with the
new inverse mapping for the IP address and names served by dhcp. This string should include
four %s's to indicate the placement of the following information from the dhcp client:

**hostname**

Value of option 12. The value returned by the dhcp server is used, if one is supplied. Else,
if the client specified a value in this file, the client-requested value is used. If neither the
client specified a requested hostname nor the server supplied one, this exec string will not
be executed.

**domainname**

Value of option 15. The value returned by the dhcp server is used, if one is supplied. Else,
if the client specified a value in this file, the client-requested value is used. If neither the
client specified a requested hostname nor the server supplied one, a null string (" ") is
supplied by dhcp. This may cause the update of address records to fail.

**Ip Address**

IP address leased to this client by the server. The string is supplied in dotted notation, for
example, 9.2.23.43.

**leasetime**

Lease time granted by the server. This string is a decimal number representing the
number of seconds of the lease.

These values are output by dhcp in this order:
hostname domainname Ip Address leasetime

A script /usr/sbin/dhcpaction has been provided with this function, as well as actions to help NIM
interact with DHCP clients. Run the script as follows:

```
/usr/sbin/dhcpaction hostname domainname ipaddress leasetime < A | PTR | BOTH | NONE > < NIM | NONIM >
```

The first four parameters are what will be used to update the DNS server. The fifth parameter tells
dhcpaction to update the A record, the PTR record, or both, or none. The options are A, PTR,
BOTH, NONE. The sixth parameter is used to tell servers that NIM is being used, and processing
needs to be done when a client changes address. The options for this are NIM and NONIM.

An example follows:

```
updateDNS "/usr/sbin/dhcpaction %s %s %s %s %s PTR
NONIM 2>&1 >>/tmp/updns.out"
```
Examples

1. In this example, we are setting up a server with a default lease time of 30 minutes. This means that any address that doesn't explicitly have a lease time set in a network, class, client, or subnet scope, will get 30 minutes. We are also setting the time between server address expiration checks to 3 minutes. This means that every 3 minutes, the server will check to see if an address has expired and mark it as expired. We are also saying the server should accept BOOTP requests and accept any client that matches the normal address assignment scheme. The normal address assignment scheme means that an address and options are assigned based on the network/subnet that the client is on.

We are also setting up two global options that should apply to all clients we serve. We are saying that there is a printer at 10.11.12.13 for everyone to use and the global domain name is dreampark. We are defining one network that has subnetting on the first 24 bits.

Thus, the network we are defining has some number of subnets and all the subnets we are specifying in this network scope have netmask of 255.255.255.0. Under that network, we are defining some options for that network and some subnets. The subnets define the actual addresses available for distribution. There are two subnets. Inside the second subnet, there is a class. The class information only applies to hosts on the second subnet that request that class. If that class is asked for the host, it will get two netbios options. If the address is in the first subnet, it will get the options in the subnet clause, which are nothing. If the host is in the second subnet, it will get all the options in the clause for the second subnet. If it also has the class, it will get the class options. If options are repeated with the same scope or a sub-scope, these options are concatenated together and set as one option. All hosts given an address from one of the two subnets will receive the options that are in the network scope.

```
leaseTimeDefault 30 minutes
leaseExpireInterval 3 minutes
supportBOOTP yes
supportUnlistedClients yes

option 9 10.11.12.13 # printer for all
option 15 dreampark # domain name

network 9.0.0.0 24
{
  subnet 9.2.218.0 9.2.218.1-9.2.218.128
  subnet 9.67.112.0 9.67.112.1-9.67.112.64
  {
    option 28 9.67.112.127 # broadcast address
    option 9 9.67.112.1 # printer 1
    option 9 9.67.112.2 # printer 2
    option 15 sandbox. # domain name
    class netbios_host
    {
      #Netbios over tcp/ip name server
      option 44 9.67.112.125
      Netbios over tcp/ip node type
      option 46 2
    }
  }
  option 15 toyland # domain name
  option 9 9.68.111.128 # printer 3
  option 33 1.2.3.4:9.8.7.1 # route to the moon
  option 33 5.6.7.8:9.8.7.2 # route to the mars
  # routes to black holes
  option 3 11.22.33.44 55.66.77.88
}
```

2. In this example, we see the output of the dhcpconf command. This format is more used by the dhcpconf GUI to store information. This format allows for multiple configurations. The dhcpconf GUI can in turn generate the specific server files for an individual server. The file specifies two of DHCP Servers, Greg and Fred. Each contain the definitions for the two servers. The dhcpconf
command can generate files specifically for Greg or Fred. The dhcpsconf command will also use the named resources (## sections) to display network pieces that have been named by the administrator.

The DHCP server Greg is responsible for network 9.3.145.0, subnet mask 255.255.255.192. The DHCP server Fred is responsible for network 9.3.146.128, subnet mask 255.255.255.240. Each server provides its own domain name. Other options named and unnamed may be placed in the server's configuration section.

Note: This format is used by dhcpsconf, which generates the appropriate configuration files for DHCP servers Greg and Fred.

```bash
# Named resources Section
## "Network 1 Subnet Netmask" "option 1 255.255.255.192"
## "Network 2 Subnet Netmask" "option 1 255.255.255.240"
## "Network 1 Domain Name" "option 15 "bizarro.austin.ibm.com"
## "Network 2 Domain Name" "option 15 "superman.austin.ibm.com"
## "Network 1 Network" "network 9.3.145.0 26"
## "Network 2 Network" "network 9.3.146.128 27"

### "DHCP Server Greg" "logItem SYSERR"
### "DHCP Server Greg" "numlogfile 6"
### "DHCP Server Greg" "logfilesize 100"
### "DHCP Server Greg" "logfile /usr/tmp/dhcpgreg.log"
### "DHCP Server Greg" "network 9.3.145.0 26"
### "DHCP Server Greg" "["
### "DHCP Server Greg" "option 15 "bizarro.austin.ibm.com"
### "DHCP Server Greg" "]"
### "DHCP Server Fred" "logItem SYSERR"
### "DHCP Server Fred" "logItem OBJERR"
### "DHCP Server Fred" "numlogfile 3"
### "DHCP Server Fred" "logfilesize 50"
### "DHCP Server Fred" "logfile /usr/tmp/dhcpsfred.log"
### "DHCP Server Fred" "network 9.3.146.128 27"
### "DHCP Server Fred" "["
### "DHCP Server Fred" "option 15 "superman.austin.ibm.com"
### "DHCP Server Fred" "]"
```

Related Information
The dhcpsd daemon, the dhcpsconf command

The DHCP Client Configuration File

TCP/IP Address and Parameter Assignment - Dynamic Host Configuration Protocol in Networks and communication management.

Problems with Dynamic Host Configuration Protocol (DHCP) in Networks and communication management.

depend File

Purpose
Describes the format of a software dependencies file.

Description
The depend file is an ASCII file used to specify information concerning software dependencies for a particular package. The file is created by a software developer.
Each entry in the `depend` file describes a single software package. The instance of the package is described after the entry line by giving the package architecture and/or version. The format of each entry and subsequent instance definition is:

```
type pkg name
```

The fields are:

<table>
<thead>
<tr>
<th>Entry</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>Defines the dependency type. This must be one of the following:</td>
</tr>
<tr>
<td>P</td>
<td>Indicates a prerequisite for installation, for example, the referenced package or versions must be installed.</td>
</tr>
<tr>
<td>I</td>
<td>Implies that the existence of the indicated package or version is incompatible. See also the X tag.</td>
</tr>
<tr>
<td>X</td>
<td>Implies that the existence of the indicated package or version is incompatible. This tag should be used instead of the I tag.</td>
</tr>
<tr>
<td>R</td>
<td>Indicates a reverse dependency. Instead of defining the packages own dependencies, this designates that another package depends on this one. This type should be used only when an old package does not have a depend file but it relies on the newer package nonetheless. Therefore, the present package should not be removed if the designated old package is still on the system since, if it is removed, the old package will no longer work.</td>
</tr>
<tr>
<td>S</td>
<td>Indicates a superseding dependency. It should be used when an earlier package has been superseded by the current package.</td>
</tr>
</tbody>
</table>

`pkg` Indicates the package abbreviation.  
`name` Specifies the full package name.

Dependency checks may be disabled using the `admin` file.

**Examples**

Shown below is an example of a `depend` file (for the NFS package):

```
P base Base System
P nsu Networking Support Utilities
P inet Internet Utilities
P rpc Remote Procedure Call Utilities
P dfs Distributed File System Utilities
```

**Related Information**

The `admin` file format, `compver` file format.

---

**dir File**

**Purpose**

Describes the format of a directory.

**Syntax**

```
#include <sys/dir.h>
```

**Description**

A directory is a file that contains information and structures necessary to define a file hierarchy. A file is interpreted as a directory by the system if it has the S_IFDIR file mode. All modifications to the structure of a directory must be performed under the control of the operating system.
The directory file format accommodates component names of up to 256 characters. This is accomplished through the use of a variable-length structure to describe individual directory entries. The structure of a directory entry follows.

**Note:** This structure is a file system-specific data structure. It is recommended that file system-independent application programs use the file system-independent `direct` structure and its associated library support routines.

```
struct direct {
    ino_t    d_ino;
    ushort   d_reclen;
    ushort   d_namelen;
    char d_name[256];
};
```

By convention, the first two entries in each directory are `. (dot)` and `.. (dot dot)`. The `. (dot)` is an entry for the directory itself. The `.. (dot dot)` entry is for the parent directory. Within the root (`/`) directory the meaning of `.. (dot dot)` is modified; because there is no parent directory, the `.. (dot dot)` entry has the same meaning as the `. (dot)` entry.

The `DIRSZ (dp)` macro gives the amount of space required to represent a directory entry. The `dp` argument is a pointer to a `direct` structure.

**Related Information**

The `dirent.h` file, `filsys.h` file, `inode` file.

The `opendir, readdir, telldir, seekdir, rewindir, or closedir` subroutine.

File systems in *Operating system and device management.*

Directories in *Operating system and device management.*

Files in *Operating system and device management.*

---

**dpid2.conf File**

**Purpose**

Defines the configuration parameters for the `dpid2` daemon.

**Description**

The `dpid2.conf` file provides the configuration information for the `dpid2` daemon. Stop the daemon using the `stopsrc` command and start it using the `startsrc` command if there are changes to the configuration file. The daemon should be under the System Resource Control (SRC) for issuing these start or stop commands.

**Keywords**

The directives are specified in the form `<keyword>=<value>`. The following keyword is case-insensitive:

- `dpiTimeOut`
  - Specifies the timeout in seconds that the `dpid2` daemon waits for the arrival of data from subagents. The default value is 8 seconds. The maximum timeout value is 600 seconds. If you specify any value less than or equal to zero, the default value is taken for processing.

**Note:** This sample configuration file is in the `/usr/samples/snmpd/dpi2` directory. You must copy the configuration file to the `/etc` directory to use the configurable parameter `dpiTimeOut`. 

---

60  AIX 5L Version 5.3 Files Reference
Examples
The following example sets the configurable parameter `dpiTimeOut` to the maximum value of 600 seconds.
dpiTimeOut=600

Files
`/etc/dpid2.conf` Defines the configuration parameters for the `dpid2` daemon.

Related Information
The `dpid2` daemon.

**dsinfo File**

**Purpose**
Contains the terminal descriptions for the Dynamic Screen utility.

**Description**
The `dsinfo` file is a database of terminal descriptions used by the Dynamic Screen utility. A terminal description typically contains the following configuration information:
- Keys defined for specific use with the Dynamic Screen utility and their function
- Number of pages of screen memory available to the terminal
- Code sequences that must be sent or received to access and use Dynamic Screen features

The `dscreen` command reads the appropriate configuration information from the `dsinfo` file to start the Dynamic Screen utility.

**Entry Format**
Line entries in the `dsinfo` file consist of a number of definition fields separated by commas. The first-line field entries are alternate screen names for the terminal. The screen name fields are separated by a | (pipe symbol).

Other line fields are strings describing the capabilities of the terminal definition to the Dynamic Screen utility. The following escape codes are recognized within these strings:

<table>
<thead>
<tr>
<th>Escape Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>\E</code>, <code>\e</code></td>
<td>Escape</td>
</tr>
<tr>
<td><code>\n</code>, <code>\l</code></td>
<td>New line</td>
</tr>
<tr>
<td><code>\r</code></td>
<td>Carriage return</td>
</tr>
<tr>
<td><code>\t</code></td>
<td>Tab</td>
</tr>
<tr>
<td><code>\b</code></td>
<td>Backspace</td>
</tr>
<tr>
<td><code>\f</code></td>
<td>Form feed</td>
</tr>
<tr>
<td><code>\s</code></td>
<td>Space</td>
</tr>
<tr>
<td><code>\n\nnn</code></td>
<td>Character with octal value</td>
</tr>
<tr>
<td><code>^x</code></td>
<td>Ctrl-x for any appropriate x.</td>
</tr>
</tbody>
</table>

Any other character preceded by a \ (backslash) yields the character itself.

Strings must be entered as the `type=string` parameter, where `type` is the string type and `string` is the string value.

If information is not entered into a string field, a comma is still used to designate the existence of the field.
**String Types and String Values**

The following string types are available:

<table>
<thead>
<tr>
<th>String Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>dskx</strong></td>
<td>Describes the action assigned to a key. This string type contains 4 characters. The 4th character indicates the action to be taken when the keystroke is received by the screen:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Key Type</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>dskb</td>
<td>Block input and output.</td>
</tr>
<tr>
<td>dskc</td>
<td>Start a new screen.</td>
</tr>
<tr>
<td>dske</td>
<td>End the <strong>Dynamic Screen</strong> utility (exit code 0).</td>
</tr>
<tr>
<td>dski</td>
<td>List keys and actions.</td>
</tr>
<tr>
<td>dskp</td>
<td>Switch to previous screen.</td>
</tr>
<tr>
<td>dsks</td>
<td>Select a specific screen.</td>
</tr>
<tr>
<td>dske</td>
<td>End the <strong>Dynamic Screen</strong> utility (exit code 0).</td>
</tr>
<tr>
<td>dskp</td>
<td>Switch to previous screen.</td>
</tr>
<tr>
<td>dsks</td>
<td>Select a specific screen.</td>
</tr>
</tbody>
</table>

Currently, the only valid **dsk** string type endings are b, c, e, l, p, q, and s. Any other key definitions used at this time are interpreted as null values and cause no internal Dynamic Screen action for the terminal definition. Other keys may be assigned values within the **Dynamic Screen** utility at a later time.

**Note:** The **dskn** string type (n for null or no operation) is guaranteed not to be used for any function assignments in future versions. It is recommended that the **dskn** string type be used instead of other null characters when no internal Dynamic Screen action is desired for a terminal definition.

The value string for each **dskx** string type has three substrings, separated by a | (pipe symbol). (To include a | in one of the substrings, use \| [backslash, pipe symbol].)

The first substring is the sequence of characters the terminal sends when the key is pressed. The second substring is a label for the key as displayed in the key listing (for example, the Shift-F1 key sequence). The third substring is a sequence of characters the **Dynamic Screen** utility sends to the terminal when the key is pressed, before performing the requested action.

| dsp | Describes a physical screen in the terminal. A **dsp** string type must be present for each physical screen in the terminal. |

The value string for each physical screen has two substrings, separated by a | (pipe symbol). (To include a | in one of the substrings, use \| [backslash, pipe symbol].)

The first substring is the sequence of characters to send to the terminal to display and output to the particular named physical page on the terminal. The second substring is usually set to clear the screen sequence. It is sent under the following two conditions:

- The creation of new terminal session
- More terminals are running than there are physical screens.

If your selection of a terminal causes the **Dynamic Screen** utility to reuse one of the physical screens, the clear-the-screen sequence is sent to the screen to indicate that the screen content does not match the output of the terminal connected to it.

**Note:** Running with more terminals than there are physical screens is not recommended. Avoid this situation by defining no more screen selection keys (**dsks=**) than physical screens (**dsp=**).
<table>
<thead>
<tr>
<th>Key Type</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>dst</td>
<td>Adjusts the Dynamic Screen utility's input timeout. The value of the string must be a decimal number. The timeout value is in tenths of a second and has a maximum value of 255. The default timeout value is 1, or one tenth of a second. When the Dynamic Screen utility recognizes a prefix of an input sequence but has not yet received all the characters in the sequence, it waits for more characters. If the timeout occurs before more characters are received, the received characters are sent to the screen, and the Dynamic Screen utility does not consider these characters as part of an input key sequence. Consider increasing the value of the dsp string if one or more of the keys to which the utility has to respond is actually a number of key combinations (for example, &lt;Ctrl-Z&gt; 1, &lt;Ctrl-Z&gt; 2, &lt;Ctrl-Z&gt; 3, and so on, for screen selection, or &lt;Ctrl-Z&gt; N, for new screen).</td>
</tr>
</tbody>
</table>

Examples

1. The following dsinfo entry describes a WYSE 60 terminal with three screens:

   wy60|wyse60|wype model 60,
   dsks=\'A`^M|Shift-F1,
   dsks=\'Aa^M|Shift-F2,
   dsks=\'Ab^M|Shift-F3,
   dsks=\'00|Ctrl-F1,
   dsks=\'01|Ctrl-F2\|\Em0\|\Em+, 
   dsks=\'02|Ctrl-F3,
   dsp=\Em0\|\Em+, dsp=\Em1\|\Em+, 

   The <Shift-F1> through <Shift-F3> key combinations are used for selecting screens 1 through 3. <Ctrl-F1> creates a new screen. <Ctrl-F2> sends the key sequence <Esc> w 0 <Esc> + to the screen. As a result, the terminal switches to window 0, the screen is cleared, and the Dynamic Screen utility ends. <Ctrl-F3> lists the keys and their functions. The three physical screens are displayed by sending the key sequences <Esc> w 0 , <Esc> w 1, and <Esc> w 2, respectively. Each time a physical screen is used for a new screen the <Esc> + key sequence is sent to the terminal to clear the screen.

2. The following dsinfo entry describes a WYSE 60 terminal with three screens, one of which is on a second computer communicating through the second serial port on the terminal. The Dynamic Screen utility must be run on both computers, with terminal type WY60-1 on the first computer and terminal type WY60-2 on the second computer (to do so specify the -t flag in the dscreen command).

   wy60-1|wyse60-1|wyse model 60 - first serial port
   dsks=\'A`^M|Shift-F1,
   dsks=\'Aa^M|Shift-F2,
   dsks=\'Ab^M|Shift-F3\|\Ed#^Ab\|\Em+, 
   dsks=\'00|Ctrl-F1,
   dsks=\'01|Ctrl-F2\|\Ed#201|\Em0\|\Em+, 
   dsks=\'02|Ctrl-F3,
   dsp=\Em0\|\Em+, dsp=\Em1\|\Em+, 

   The first two key entries for terminal type WY60-1 are identical to the entry in example 1. The third key entry, of type dsksb, specifies that input and output are blocked when the <Esc> d # <Ctrl-A> b <CR>
<Ctrl-T> <Esc> e 9 key sequence is sent to the terminal. As a result, output is blocked, and the
Dynamic Screen utility continues to scan input for key sequences but discards
all other input. The <Esc> d # sequence puts the terminal in transparent print mode, which echoes all
keystrokes up to <Ctrl-T> out the other serial port. The <Ctrl-A> b <CR> key sequence is sent out to
the other serial port, informing the Dynamic Screen utility on the second computer that it should
activate the window associated with the <Shift-F3> key. The <Ctrl-T> key sequence takes the terminal
out of transparent print mode, and the <Esc> e 9 key sequence informs the terminal to switch to the
other serial port for data communications.

The other computer takes over and sends the <Esc> w 2 key sequence to switch to the third physical
screen and then resumes normal communication.

The WY60-2 entry follows the same general pattern for the <Shift-F1> and <Shift-F2> key
combinations, which switch to transparent print mode, send a function key string to the other computer,
switch transparent print off, and switch to the other serial port.

The end key <Ctrl-F2> works the same for both computers. It sends the end key sequence to the other
computer through the transparent print mechanism, switches the terminal to window 0, clears the
screen, and exits.

Files

/etc/dsinfo Contains the terminal descriptions for the Dynamic Screen utility.

Related Information

The dscreen command.

dumpdates File

Purpose

Describes the format of the dumpdates file.

Description

The /etc/dumpdates file holds filesystem backup information for the backup and rdump commands. The
dumpdates file is maintained by using the -u option when performing file system backups. The following is
the dumpdates data structure:

```
struct idates {
  char id_name[MAXNAMLEN+3];
  char id_incno;
  time_t id_ddate;
}
```

The struct idates describes an entry in the /etc/dumpdates file where the backup history is kept. The
fields of the structure are:

- id_name The name of the file system.
- id_incno The level number of the last backup.
- id_ddate The date of the incremental backup in system format.
- MAXNAMLEN The maximum value of this variable is 255.

Files

/etc/dumpdates Specifies the path name of the symbolic link to the dumpdates file.
Related Information

The **backup** command, **rdump** command.

Backup methods in *Operating system and device management.*

---

**e789_ctbl File for HCON**

**Purpose**

Contains the default binary color definition table for HCON.

**Description**

The `/usr/lib/hcon/e789_ctbl` file contains the default color definition table for the Host Connection Program (HCON) in binary form.

Instances of the `e789_ctbl` file can also occur in user $HOME directories. The color definition table can be customized using the **hconutil** command. If the user issuing the **hconutil** command does not specify a name for the new table, the command names the `e789_ctbl` table and places it in the user $HOME directory. To use a customized table, an HCON user must specify the file name of the table in an HCON session profile.

**Files**

```
/usr/lib/hcon/e789_ctbl
```

Specifies the path of the `e789_ctbl` file.

---

**Related Information**

The **chhcons** command.

---

**e789_ktbl File for HCON**

**Purpose**

Contains the default binary keyboard definition table used by HCON.

**Description**

The `/usr/lib/hcon/e789_ktbl` file contains the default keyboard definition table used by the Host Connection Program (HCON) in binary form.

HCON key names are mapped to specific keys on each supported keyboard. The HCON emulator program uses these key mappings to generate the correct key function on all the supported keyboards. HCON key mappings can be customized using the **hconutil** command.

Instances of the `e789_ktbl` file can also occur in user $HOME directories. The keyboard definition table can be customized using the **hconutil** command. If the user issuing the **hconutil** command does not specify a name for the new table, the command names the `e789_ktbl` table and places it in the user $HOME directory. To use a customized table, an HCON user must specify the file name of the table in an HCON session profile.

**Files**

```
/usr/lib/hcon/e789_ktbl
```

Specifies the path of the `e789_ktbl` file.
Related Information
The `chhcons` command.

eimadmin.conf File

Purpose
Stores system Enterprise Identity Mapping (EIM) connection information from the `eimadmin` command.

Description
This file is used to store system Enterprise Identity Mapping (EIM) connection information from the `eimadmin` command. Use the `eimadmin` command to create and update this file. The connection information stored by the `eimadmin.conf` file includes the EIM domain and its controlling server, the identity with which to authenticate (bind) to the server, and the authentication method.

The meanings of the `eimadmin.conf` file's fields are as follows:

- **LdapURL**: Specifies the URL and port for the LDAP server controlling the EIM data. This field takes the following format:
  - ldap://some.ldap.host:389
  - ldaps://secure.ldap.host:636

- **KerberosRegistry**: Specifies the name of a Kerberos registry.

- **LocalRegistry**: Specifies the full distinguished name (DN) of the EIM domain. This name begins with `ibm-eimDomainName=` and consists of the following elements:
  - **domainName**: The name of the EIM domain you are creating. For example, `MyDomain`.
  - **parent distinguished name**: The distinguished name for the entry immediately above the given entry in the directory information tree hierarchy, such as `o=ibm,c=us`. For example:
    - `ibm-eimDomainName=MyDomain,o=ibm,c=us`

- **ConnectionMethod**: Specifies the method of authentication to the LDAP server. You can select one of the following methods:
  - SIMPLE (bind DN and password). (DEFAULT method).
  - CRAM-MD5 (bind DN and protected password).
  - EXTERNAL (digital certificate).
  - GSSAPI (Kerberos). Uses the default Kerberos credential. The credential must be established using a service such as `kinit` before running EIM.

- **BindDn**: The distinguished name to use for the simple bind to LDAP. For example, `cn=admin`. The bind distinguished name has one of the following EIM authorities:
  - EIM administrator
  - EIM registries administrator
  - EIM registry X administrator
  - EIM identifiers administrator

- **BindPassword**: Specifies the password associated with the bind DN.

- **SSLKeyFile**: The name of the SSL key database file, including the full path name. If the file cannot be found, the name of a RACF key ring that contains authentication certificates is used. This value is required for SSL communications with a secure LDAP host. For example:
  - `/u/eimuser/ldap.kdb`
SSLKeyPassword  The password required to access the encrypted information in the key database file. As an alternative, you can specify an SSL password stash file by prefixing the stash file name with file://. For example:
file:///u/eimuser/ldapclient.sth

SSLKeyCert  Identifies which certificate to use from the key database file or RACF key ring. If a certificate label is not specified, the default certificate in the file or ring is used.

Example

```
-> /usr/bin/eimadmin -X -d ibm-eimDomainName='ibm-eimDomainName=MyDomain,o=ibm,c=us'
   -h 'ldap://keystone.austin.ibm.com:389' -S 'SIMPLE'-b 'cn=admin' -w 'secret'
-> cat /etc/eimadmin.conf
EimConfiguration:
   LdapURL="ldap://keystone.austin.ibm.com:389"
   KerberosRegistry=""
   LocalRegistry=""
   EimDomain="ibm-eimDomainName=MyDomain,o=ibm,c=us"
   ConnectionMethod="SIMPLE-b"
   BindDn=""
   BindPassword=""
   SSLKeyFile=""
   SSLKeyPassword=""
   SSLKeyCert=""
```

Location

```
/etc/eimadmin.conf
```
Contains the eimadmin.conf file.

Related Information

The eimadmin command.

environ File

Purpose
Defines the environment attributes for users.

Description
The /etc/security/environ file is an ASCII file that contains stanzas with the environment attributes for users. Each stanza is identified by a user name and contains attributes in the Attribute=Value form, with a comma separating the attributes. Each attribute is ended by a new-line character, and each stanza is ended by an additional new-line character.

If environment attributes are not defined, the system uses default values. Each user stanza can have the following attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>usrenv</td>
<td>Defines variables to be placed in the user environment when the initial login command is given or when the su command resets the environment. The value is a list of comma-separated attributes. The default value is an empty string.</td>
</tr>
<tr>
<td>sysenv</td>
<td>Defines variables to be placed in the user protected state environment when the initial login command is given or when the su command resets the environment. These variables are protected from access by unprivileged programs so other programs can depend on their values. The default value is an empty string.</td>
</tr>
</tbody>
</table>
For a description of environment variables, refer to the `/etc/environment` file.

Access to all the user database files should be through the system commands and subroutines defined for this purpose. Access through other commands or subroutines may not be supported in future releases.

The `mkuser` command creates a user stanza in this file. The initialization of the attributes depends upon their values in the `/usr/lib/security/mkuser.default` file. The `chuser` command can change these attributes, and the `lsuser` command can display them. The `rmuser` command removes the entire record for a user.

**Security**

**Access Control:**

This command should grant read (r) access to the root user, members of the security group, and others consistent with the security policy for the system. Only the root user should have write (w) access.

**Auditing Events:**

<table>
<thead>
<tr>
<th>Event</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>S_ENVIRON_WRITE</td>
<td>file name</td>
</tr>
</tbody>
</table>

**Examples**

A typical stanza looks like the following example for user `dhs`:

```
dhs:
  usrenv = "MAIL=/home/spool/mail/dhs,MAILCHECK=600"
  sysenv = "NAME=dhs@delos"
```

**Files**

- `/etc/security/environ`
- `/etc/environment`
- `/etc/group`
- `/etc/passwd`
- `/etc/security/passwd`
- `/etc/security/user`
- `/etc/security/limits`
- `/usr/lib/security/mkuser.default`
- `/etc/security/lastlog`

Specifies the path to the file.

Specifies the basic environment for all processes.

Contains the basic attributes of groups.

Contains the extended attributes of groups.

Contains the basic attributes of users.

Contains password information.

Contains the extended attributes of users.

Contains the process resource limits of users.

Contains the default values for user accounts.

Contains last login information.

**Related Information**

The `chuser` command, `login` command, `lsuser` command, `mkuser` command, `rmuser` command, `setsenv` command, `su` command.

The `getpenv` subroutine, `getuserattr` subroutine, `putuserattr` subroutine, `setpenv` subroutine.

File and system security in *Operating system and device management*.

**environment File**

**Purpose**

Sets up the user environment.
Description

The `/etc/environment` file contains variables specifying the basic environment for all processes. When a new process begins, the `exec` subroutine makes an array of strings available that have the form `Name=Value`. This array of strings is called the environment. Each name defined by one of the strings is called an environment variable or shell variable. The `exec` subroutine allows the entire environment to be set at one time.

Environment variables are examined when a command starts running. The environment of a process is not changed by altering the `/etc/environment` file. Any processes that were started prior to the change to the `/etc/environment` file must be restarted if the change is to take effect for those processes. If the `TZ` variable is changed, the `cron` daemon must be restarted, because this variable is used to determine the current local time.

The following restrictions apply, when modifying the `environment` file:

- Ensure that newly created environment variables do not conflict with standard variables such as `MAIL`, `PS1`, `PS2`, and `IFS`.
- Ensure that the information in the `environment` file is in the `Name=Value` format. Unlike profile scripts, the `environment` file is not a shell script and does not accept data in any format other than the `Name=Value` format.

The Basic Environment

When you log in, the system sets environment variables from the `environment` file before reading your login profile, `.profile`.

The following variables make up the basic environment:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOME</td>
<td>The full path name of the user login or HOME directory. The <code>login</code> program sets this to the name specified in the <code>/etc/passwd</code> file.</td>
</tr>
<tr>
<td>LANG</td>
<td>The locale name currently in effect. The LANG variable is set in the <code>/etc/environment</code> file at installation time.</td>
</tr>
<tr>
<td>NLSPATH</td>
<td>The full path name for message catalogs. The default is:</td>
</tr>
<tr>
<td></td>
<td><code>/usr/lib/nls/msg/%L/%N:</code></td>
</tr>
<tr>
<td></td>
<td><code>/usr/lib/nls/msg/%L/%N.cat:</code></td>
</tr>
<tr>
<td></td>
<td>where %L is the value of the LC_MESSAGES category and %N is the catalog file name.</td>
</tr>
<tr>
<td>Note:</td>
<td>See the <code>chlang</code> command for more information about changing message catalogs.</td>
</tr>
<tr>
<td>LC__FASTMSG</td>
<td>If LC_FASTMEG is set to false, POSIX-compliant message handling is performed. If LC__FASTMSG is set to true, it specifies that default messages should be used for the C and POSIX locales and that NLSPATH is ignored. If this variable is set to anything other than false or unset, it is considered the same as being set to true. The default value is LC__FASTMSG=true in the <code>/etc/environment</code> file.</td>
</tr>
<tr>
<td>LOCPATH</td>
<td>The full path name of the location of National Language Support tables. The default is <code>/usr/lib/nls/loc</code> and is set in the <code>/etc/profile</code> file. If the LOCPATH variable is a null value, it assumes that the current directory contains the locale files.</td>
</tr>
<tr>
<td>Note:</td>
<td>All setuid and setgid programs will ignore the LOCPATH environment variable.</td>
</tr>
<tr>
<td>PATH</td>
<td>The sequence of directories that commands such as the <code>sh</code>, <code>time</code>, <code>nice</code> and <code>nohup</code> commands search when looking for a command whose path name is incomplete. The directory names are separated by colons.</td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>TZ</td>
<td>The time-zone information. The TZ environment variable is set by the <code>/etc/environment</code> file. The TZ environment variable has the following format (spaces inserted for readability): std offset dst offset , rule</td>
</tr>
</tbody>
</table>

The fields within the TZ environment variable are defined as follows:

**std and dst**

Designate the standard (std) and summer (dst) time zones. Only the std value along with the appropriate offset value is required. If the dst value is not specified, summer time does not apply. The values specified may be no less than three and no more than TZNAMEMAX bytes in length. The length of the variables corresponds to the %Z field of the `date` command; for libc and libbsd, TZNAMEMAX equals three characters. Any nonnumeric ASCII characters except the following may be entered into each field: a leading : (colon), a , (comma), a - (minus sign), a + (plus sign), or the ASCII null character.

Note: POSIX 1.0 reserves the leading : (colon) for an implementation-defined TZ specification. The operating system disallows the leading colon, selecting CUT0 and setting the %Z field to a null string.

An example of std and dst format is as follows: EST5EDT

**EST**

Specifies Eastern U.S. standard time.

**5**

Specifies the offset, which is 5 hours behind Coordinated Universal Time (CUT).

**EDT**

Specifies the corresponding summer time zone abbreviation.

**Note:** See "Time Zones" for a list of time zone names defined for the system.

**offset**

Denotes the value added to local time to equal Coordinated Universal Time (CUT). CUT is the international time standard that has largely replaced Greenwich Mean Time. The offset variable has the following format:

```
hh:mm:ss
```

The fields within the offset variable are defined as follows:

**hh**

Specifies the dst offset in hours. This field is required. The hh value can range between the integers -12 and +11. A negative value indicates the time zone is east of the prime meridian; a positive value or no value indicates the time zone is west of the prime meridian.
<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mm</td>
<td>Specifies the <strong>dst</strong> offset detailed to the minute. This field is optional. If the <code>mm</code> value is present, it must be specified between 0 and 59 and preceded by a : (colon).</td>
</tr>
<tr>
<td>ss</td>
<td>Specifies the <strong>dst</strong> offset detailed to the second. The <code>ss</code> field is optional. If the <code>ss</code> value is present, it must be specified between 0 and 59 and preceded by a : (colon).</td>
</tr>
</tbody>
</table>

An **offset** variable must be specified with the **std** variable. An **offset** variable for the **dst** variable is optional. If no offset is specified with the **dst** variable, the system assumes that summer time is one hour ahead of standard time.

As an example of offset syntax, Zurich is one hour ahead of CUT, so its offset is -1. Newfoundland is 1.5 hours ahead of eastern U.S. standard time zones. Its syntax can be stated as any of the following: 3:30, 03:30, +3:30, or 3:30:00.

**rule**  
The **rule** variable indicates when to change to and back from summer time. The **rule** variable has the following format:

```
start/time,end/time
```

The fields within the **rule** variable are defined as follows:

<table>
<thead>
<tr>
<th>start</th>
<th>end</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specifies the change from standard to summer time.</td>
<td>Specifies the return to standard time from summer time.</td>
<td>Specifies when the time changes occur within the time zone. For example, if the <strong>time</strong> variable is encoded for 2 a.m. then the time changes when the time zone reaches 2 a.m. on the date specified in the <strong>start</strong> variable.</td>
</tr>
</tbody>
</table>
Variable / Description
Delimits the start date, end date, and time variables.

, (Comma) Delimits two date and time pairs.

The start and end variables support a syntax for Julian time (J) and a syntax for leap years (M):

Jn
Mm.n.d

In the J syntax, the n variable has the value of 1 through 365. Leap days are not counted. In the M syntax, m is the month, n the week, and d the day of the week starting from day 0 (Sunday).

The rule variable has the same format as the offset variable except no leading - (minus sign) or + (plus sign) is allowed. The default of the start variable is 02:00:00 (2 a.m.).

Note: The time zone offsets and time change points are interrelated and context-dependent. The rule variable’s runtime execution semantics change as a function of the offsets. For example, if the summer time zone changes one hour, as in CST6CDT5, (the default 2 a.m.) summer time changes instantaneously from 2 a.m. to 3 a.m. CDT. The fall change is from 2 a.m. CDT to 1 a.m. CST. The respective changes for a time zone of CST6CDT4 are 2 a.m. CST to 4 a.m. CDT and 2 a.m. CDT to 12 a.m. CST.

In an example of the rule variable, if the law changed so that the Central United States experienced summer time between Julian 129 and Julian 131, the TZ variable would be stated as follows:

TZ=CST6CDT5,J129,J131

In this example, the dates indicated are May 09 and May 11,1993, respectively. (Use the date +%j command to get the Julian date number.)

In another example, if the time changes were to occur at 2 a.m. CST and 19:30 CDT, respectively, the variables would be stated as follows:
TZ=CST6CDT5,J129,J131/19:30

In nonleap years, the fallback time change would be from 19:30 CDT to 18:30 CST on May 11 (1993).

For the leap year (M) syntax, the spring ahead date would be 2 May and the fallback date is 9 May. The variables are stated as follows:
TZ=CST6CDT5,M5.1.0,M5.2.0

Time Zones
The system defines the following time zones and time zone names:

Note: Coordinated Universal Time (CUT) is the international time standard.

Table 2. Time Zones Defined on the System

<table>
<thead>
<tr>
<th>Name</th>
<th>Time Zone</th>
<th>CUT Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUT0GDT</td>
<td>Coordinated Universal Time</td>
<td>CUT</td>
</tr>
<tr>
<td>GMT0BST</td>
<td>United Kingdom</td>
<td>CUT</td>
</tr>
<tr>
<td>WET0WEST</td>
<td>Western Europe</td>
<td>CUT</td>
</tr>
<tr>
<td>AZOREST1AZORED</td>
<td>Azores, Cape Verde</td>
<td>CUT -1</td>
</tr>
<tr>
<td>FALKST2FALKD</td>
<td>Falkland Islands</td>
<td>CUT -2</td>
</tr>
<tr>
<td>GRNLDNDS3GRNLDNDDT</td>
<td>Greenland, East Brazil</td>
<td>CUT -3</td>
</tr>
</tbody>
</table>
Table 2. Time Zones Defined on the System (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Time Zone</th>
<th>CUT Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>AST4ADT</td>
<td>Central Brazil</td>
<td>CUT -4</td>
</tr>
<tr>
<td>EST5EDT</td>
<td>Eastern United States, Colombia</td>
<td>CUT -5</td>
</tr>
<tr>
<td>CST6CDT</td>
<td>Central United States, Honduras</td>
<td>CUT -6</td>
</tr>
<tr>
<td>MST7MDT</td>
<td>Mountain United States</td>
<td>CUT -7</td>
</tr>
<tr>
<td>PST8MDT</td>
<td>Pacific United States, Yukon</td>
<td>CUT -8</td>
</tr>
<tr>
<td>AST9ADT</td>
<td>Alaska</td>
<td>CUT -9</td>
</tr>
<tr>
<td>HST10HDT</td>
<td>Hawaii, Aleutian Islands</td>
<td>CUT -10</td>
</tr>
<tr>
<td>BST11BDT</td>
<td>Bering Strait</td>
<td>CUT -11</td>
</tr>
<tr>
<td>NZST-12NZDT</td>
<td>New Zealand</td>
<td>CUT +12</td>
</tr>
<tr>
<td>MET-11METDT</td>
<td>Solomon Islands</td>
<td>CUT +11</td>
</tr>
<tr>
<td>EET-10EETDT</td>
<td>Eastern Australia</td>
<td>CUT +10</td>
</tr>
<tr>
<td>JST-9JSTDT</td>
<td>Japan</td>
<td>CUT +9</td>
</tr>
<tr>
<td>KORST-9KORDT</td>
<td>Korea</td>
<td>CUT +9</td>
</tr>
<tr>
<td>WAUST-8WAUDT</td>
<td>Western Australia</td>
<td>CUT +8</td>
</tr>
<tr>
<td>TAIST-8TAIDT</td>
<td>Taiwan</td>
<td>CUT +8</td>
</tr>
<tr>
<td>THAIST-7THAIDT</td>
<td>Thailand</td>
<td>CUT +7</td>
</tr>
<tr>
<td>TASHST-6TASHDT</td>
<td>Central Asia</td>
<td>CUT +6</td>
</tr>
<tr>
<td>PAKST-5PAKDT</td>
<td>Pakistan</td>
<td>CUT +5</td>
</tr>
<tr>
<td>WST-4WDT</td>
<td>Gorki, Central Asia, Oman</td>
<td>CUT +4</td>
</tr>
<tr>
<td>MEST-3MEDT</td>
<td>Turkey</td>
<td>CUT +3</td>
</tr>
<tr>
<td>SAUST-3SAUDT</td>
<td>Saudi Arabia</td>
<td>CUT +3</td>
</tr>
<tr>
<td>EET-2EEST</td>
<td>Eastern Europe</td>
<td>CUT +2</td>
</tr>
<tr>
<td>USAST-2USADT</td>
<td>South Africa</td>
<td>CUT +2</td>
</tr>
<tr>
<td>CET-1CEST</td>
<td>Central Europe</td>
<td>CUT +1</td>
</tr>
</tbody>
</table>

Files

/etc/profile
Specifies variables to be added to the environment by the shell.

/etc/environment
Specifies the basic environment for all processes.

$HOME/.profile
Specifies the environment for specific user needs.

/etc/passwd
Specifies user IDs.

Related Information

The at command, chlang command, env command, getty command, login command, sh command.

The exec subroutine, getenv subroutine.

errors File for BNU

Purpose
Contains a record of uucico daemon errors.
Description
The `/var/spool/uucp/.Admin/errors` file contains a record of `uucico` daemon errors that the Basic Networking Utilities (BNU) program cannot correct. For example, if the `uucico` daemon is unable to access a directory that is needed for a file transfer, the BNU program records this in the `errors` file.

If debugging is enabled for the `uucico` daemon, the BNU program sends the error messages to standard output instead of to the `errors` file.

Examples
The text of an error which might appear in the `errors` file is:

```
ASSERT ERROR (uucico) pid: 303 (7/18-8:25:09) SYSTAT OPEN FAIL /v
ar/spool/uucp/.Status/ (21) [SCCSID: @(#)systat.c 7.2 87/07/08
16:43:37, FILE: systat.c, LINE:100]
```

This error occurred on July 18 at 8:25:09 a.m. [(7/18-8:25:09)] when the `uucico` daemon, running as process 303 [(uucico) pid: 303], could not open the `/var/spool/uucp/.Status` directory [SYSTAT OPEN FAIL /var/spool/uucp/.Status/]. To prevent this error from occurring again, you should make sure the permissions for the .Status directory are correct. It should be owned by the `uucp` login ID and group `uucp`, with permissions of 777 (read, write, and execute for owner, group, and all others).

Files
- `/var/spool/uucp/.Admin` directory
  - Contains the `errors` file and other BNU administrative files.
- `/var/spool/uucp/.Status/` `SystemName`
  - Lists the last time a remote system was contacted and the minimum time until the next retry.
  - Specifies the path of the `errors` file.

Related Information
- The `uudemon.cleanu` command.
- The `uucico` daemon.
- BNU File and Directory Structure and BNU maintenance in *Networks and communication management*.

*ethers* File for NIS

Purpose
Contains the Ethernet addresses of hosts on the Internet network.

Description
The `/etc/ethers` file contains information regarding the known (48-bit) Ethernet addresses of hosts on the Internet. The file contains an entry for each host. Each entry consists of the following information:

- Ethernet address
- Official host name

Items are separated by any number of blanks or tab characters. A # (pound sign) indicates the beginning of a comment that extends to the end of the line.

The standard form for Ethernet addresses is `x:x:x:x:x:x` where `x` is a hexadecimal number between 0 and ff, representing one byte. The address bytes are always in network order. Host names may contain
any printable character other than a space, tab, new line, or comment character. It is intended that host names in the /etc/ethers file correspond to the host names in the /etc/hosts file.

This file is part of NFS in Network Support Facilities.

Files

/etc/ethers Specifies the path of the ethers file.
/etc/hosts Contains Internet addresses.

Related Information

The [etc/hosts] file.

NFS Services in Networks and communication management.

Network Information Service Overview in Networks and communication management.

events File

Purpose

Contains information about system audit events.

Description

The /etc/security/audit/events file is an ASCII stanza file that contains information about audit events. The file contains just one stanza, auditpr, which lists all the audit events in the system. The stanza also contains formatting information that the auditpr command needs to write an audit tail for each event.

Each attribute in the stanza is the name of an audit event, with the following format:

AuditEvent = FormatCommand

The format command can have the following parameters:

- Parameter (empty) Description
  - The event has no tail.

- Parameter printf Format Description
  - The tail is formatted according to the string supplied for the Format parameter. The %x symbols within the string indicate places for the audit trail to supply data.

- Parameter Program -i n Arg ... Description
  - The tail is formatted by the program specified by the Program parameter. The -i n parameter is passed to the program as its first parameter, indicating that the output is to be indented by n spaces. Other formatting information can be specified with the Arg parameter. The audit event name is passed as the last parameter. The tail is written to the standard input of the program.

Audit Event Formatting Information

<table>
<thead>
<tr>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>%A</td>
<td>Formatted output is similar to the aclget command.</td>
</tr>
<tr>
<td>%c</td>
<td>Format a single byte as a character.</td>
</tr>
<tr>
<td>%D</td>
<td>Formatted as a device major and minor number.</td>
</tr>
<tr>
<td>%d</td>
<td>Formatted as a 32-bit signed decimal integer</td>
</tr>
<tr>
<td>Format</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>%G</td>
<td>Formatted as a comma-separated list of group names or numerical identifiers.</td>
</tr>
<tr>
<td>%L</td>
<td>Formatted as a text string which describes the identity associated with an Internet socket and the socket itself.</td>
</tr>
<tr>
<td>%d</td>
<td>Formatted as a 64-bit signed decimal integer</td>
</tr>
<tr>
<td>%o</td>
<td>Formatted as a 64-bit octal value.</td>
</tr>
<tr>
<td>%x</td>
<td>Format as a 64-bit hexadecimal value.</td>
</tr>
<tr>
<td>%X</td>
<td>Formatted as a 64-bit hexadecimal value with uppercase letters.</td>
</tr>
<tr>
<td>%o</td>
<td>Formatted as 32-bit octal integer.</td>
</tr>
<tr>
<td>%l</td>
<td>Formatted output is similar to the pclget command.</td>
</tr>
<tr>
<td>%s</td>
<td>Formatted as a text string which describes an Internet socket.</td>
</tr>
<tr>
<td>%s</td>
<td>Formatted as a text string.</td>
</tr>
<tr>
<td>%T</td>
<td>Formatted as a text string giving include date and time with 6 significant digits for the seconds DD Mmm YYYY HH:MM:SS:mmmuu).</td>
</tr>
<tr>
<td>%u</td>
<td>Formatted as a 32-bit unsigned integer.</td>
</tr>
<tr>
<td>%x</td>
<td>Formatted as a 32-bit hexadecimal integer.</td>
</tr>
<tr>
<td>%X</td>
<td>Formatted as a 32-bit hexadecimal integer with uppercase letters.</td>
</tr>
<tr>
<td>%%</td>
<td>A single ‘%’ character.</td>
</tr>
</tbody>
</table>

**Security**

Access Control: This file should grant read (r) access to the root user and members of the audit group, and grant write (w) access only to the root user.

**Examples**

To format the tail of an audit record for new audit events, such as FILE_Open and PROC_Create, add format specifications like the following to the auditpr stanza in the /etc/security/audit/events file:

```plaintext
auditpr:
    FILE_Open = printf "flags: %d mode: %o \n    fd: %d filename: %s"
    PROC_Create = printf "forked child process %d"
```

**Files**

/etc/security/audit/events
- Specifies the path to the file.

/etc/security/audit/config
- Contains audit system configuration information.

/etc/security/audit/objects
- Contains information about audited objects.

/etc/security/audit/bincmds
- Contains auditbin backend commands.

/etc/security/audit/streamcmds
- Contains auditstream commands.

**Related Information**

The audit command, auditpr command.

[Setting Up Auditing](#) in *Operating system and device management*.

[Auditing overview](#) and [Security Administration](#) in *Operating system and device management*.
Execute (X.*) Files for BNU

Purpose
Contains instructions for running commands that require the resources of a remote system.

Description
The execute (X.*) files of the Basic Networking Utilities (BNU) contain instructions for running commands that require the resources of a remote system. They are created by the uux command.

The full path name of a uux command execute file is a form of the following:

/var/spool/uucp/SystemName/X.RemoteSystemNxxxx

where the SystemName directory is named for the local computer and the RemoteSystem directory is named for the remote system. The N character represents the grade of the work, and the xxxx notation is the four-digit hexadecimal transfer-sequence number; for example, X.zeusN2121.

Note: The grade of the work specifies when the file is to be transmitted during a particular connection. The grade notation is a single number (0-9) or letter (A-Z, a-z). Lower sequence characters cause the file to be transmitted earlier in the connection than do higher sequence characters. The number 0 is the highest grade, signifying the earliest transmittal; z is the lowest grade, specifying the latest transmittal. The default grade is N.

Standard Entries in an Execute File
An execute file consists of several lines, each with an identification character and one or more entries:

User Line:

<table>
<thead>
<tr>
<th>Identification Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>U</td>
<td>UserName SystemName Specifications the login name of the user issuing the uux command and the name of the system that issued the command.</td>
</tr>
</tbody>
</table>

Error Status Line:

<table>
<thead>
<tr>
<th>Identification Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N or Z</td>
<td>Indicates the error status.</td>
</tr>
<tr>
<td>N</td>
<td>Indicates that a failure message is not sent to the user issuing the uux command if the specified command does not execute successfully on the remote system.</td>
</tr>
<tr>
<td>Z</td>
<td>Indicates that a failure message is sent to the user issuing the uux command if the specified command does not execute successfully on the remote system.</td>
</tr>
</tbody>
</table>

Requester Name:

<table>
<thead>
<tr>
<th>Identification Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>UserName Specifications the login ID of the user requesting the remote command execution.</td>
</tr>
</tbody>
</table>
**Required File Line:**

<table>
<thead>
<tr>
<th>Identification</th>
<th>Character</th>
<th>Description</th>
</tr>
</thead>
</table>
| **F**          | FileName  | Contains the names of the files required to execute the specified command on the remote system. The `FileName` parameter can be either the complete path name of the file, including the unique transmission name assigned by the BNU program, or simply the transmission name without any path information.  

The required file line can contain zero or more file names. The `uuxqt` daemon checks for the existence of all listed files before running the specified command. |

**Standard Input Line:**

<table>
<thead>
<tr>
<th>Identification</th>
<th>Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>I</strong></td>
<td>FileName</td>
<td>Specifies the standard input to be used.</td>
</tr>
</tbody>
</table>

The standard input is either specified by a `<` (less than) symbol in the command string or inherited from the standard input of the `uux` command if that command was issued with the `-` (minus sign) flag. If standard input is specified, the input source is also listed in an **F** (Required File) line. If standard input is not specified, the BNU program uses the `/dev/null` device file.

**Standard Output Line:**

<table>
<thead>
<tr>
<th>Identification</th>
<th>Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>O</strong></td>
<td>FileName SystemName</td>
<td>Specifies the names of the file and system that are to receive standard output from the command execution. Standard output is specified by a <code>&gt;</code> (greater than) symbol within the command string. (The <code>&gt;&gt;</code> sequence is not valid in <code>uux</code> commands.) As is the case with standard input, if standard output is not specified, the BNU program uses the <code>/dev/null</code> device file.</td>
</tr>
</tbody>
</table>

**Command Line:**

<table>
<thead>
<tr>
<th>Identification</th>
<th>Character</th>
<th>Description</th>
</tr>
</thead>
</table>
| **C**          | CommandString | Gives the command string that the user requests to be run on the specified system. The BNU program checks the `/etc/uucp/Permissions` file on the designated computer to see whether the login ID can run the command on that system.  

All required files go to the execute file directory, usually `/var/spool/uucp/Xqtdir`  

After execution, the standard output is sent to the requested location. |

**Examples**

1. User `amy` on local system `zeus` issued the following command:
   
   `uux - "diff /home/amy/out hera!/home/amy/out2 > ~/DF"`
   
   The command in this example invokes the `uux` command to run a `diff` command on the local system, comparing the `/home/amy/out` file with the `/home/amy/out2` file, which is stored on remote system `hera`. The output of the comparison is placed in the `DF` file in the public directory on the local system.  

   The preceding command produces the `/var/spool/uucp/hera/X.zeusN212F` execute file, which contains the following information:
   
   The user line identifies the user `amy` on the system `zeus`. The error-status line indicates that `amy` will receive a failure status message if the `diff` command fails to execute. The requestor is `amy`, and the file required to execute the command is the following data file:
The output of the command is to be written to the public directory on the system zeus with the file name DF. (The ~ (tilde) is the shorthand way of specifying the public directory.) The final line is the command string that the user amy entered with the uux command.

2. The following is another example of an execute file:

U uucp hera
  # don't return status on failure
N
  # return address for status or input return
R uucp
  F D.hera5eb7f7b
  I D.hera5eb7f7b
  C rmail amy

   This indicates that user uucp on system hera is sending mail to user amy, who is also working on system hera.

Files

/etc/uucp/Permissions
/etc/uucp/Systems
/var/spool/uucp/SystemName directory
/var/spool/uucp/SystemName/C.*
/var/spool/uucp/Xqtdir directory
/var/spool/uucppublic/* directory

Describes access permissions for remote systems.
Describes accessible remote systems.
Contains BNU command, data, and execute files.
Contains instructions for transfers.
Contains lists of commands that remote systems are permitted to execute.
Contains transferred files.

Related Information

The diff command, uux command.
The uuxqt daemon.

BNU File and Directory Structure, BNU Daemons, and BNU maintenance commands in Networks and communication management.

exports File for NFS

Purpose
Contains a list of directories that can be exported to Network File System (NFS) clients.

Description
The /etc/exports file contains an entry for each directory that can be exported to NFS clients. This file is read automatically by the exports command. If you change this file, you must run the exports command before the changes can affect the way the daemon operates.
Only when this file is present during system startup does the `rc.nfs` script execute the `exportfs` command and start the `nfsd` and `mountd` daemons.

**Restriction:** You cannot export either a parent directory or a subdirectory of an exported directory within the same file system.

If there are two entries for the same directory with different NFS versions 2 (or 3) and NFS versions 4 in the `/etc/exports` file, the `exportfs` command exports both of the two entries.

If the options for NFS versions 2 (or 3) and 4 are the same for a directory, there can be one entry in the `/etc/exports` file specifying `-vers=3:4`.

Entries in the file are formatted as follows:

```
Directory-Option [ , Option ] ...
```

These entries are defined as follows:

**Entry** | **Definition**
--- | ---
`Directory` | Specifies the directory name.
**Entry**

**Definition**
Specifies the optional characteristics for the directory being exported. You can enter more than one variable by separating them with commas. For options taking a Client parameter, Client can specify a hostname, a dotted IP address, a network name, or a subnet designator. A subnet designator is of the form @host/mask, where host is either a hostname or a dotted IP address and mask specifies the number of bits to use when checking access. If mask is not specified, a full mask is used. For example, the designator @client.group.company.com/16 will match all Clients on the company.com subnet. A designator of @client.group.company.com/24 will match only the Clients on the group.company.com subnet. Choose from the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ro</td>
<td>Exports the directory with read-only permission. If not specified, the directory is exported with read-write permission.</td>
</tr>
<tr>
<td>ro=Client[:Client]</td>
<td>Exports the directory with read-only permission to the specified Clients. Exports the directory with read-write permissions to Clients not specified in the list. A read-only list cannot be specified if a read-write list has been specified.</td>
</tr>
<tr>
<td>rw</td>
<td>Exports the directory with read-write permission to all Clients.</td>
</tr>
<tr>
<td>rw = Client [:Client]</td>
<td>Exports the directory with read-write permission to the specified Clients. Exports the directory read-only to Clients not in the list. A read-write list cannot be specified if a read-only list has been specified.</td>
</tr>
<tr>
<td>access = Client[:Client,...]</td>
<td>Gives mount access to each Client listed. If not specified, any Client is allowed to mount the specified directory.</td>
</tr>
<tr>
<td>anon=UID</td>
<td>If a request comes from a root user, use the user identification (UID) value as the effective user ID. The default value for this option is -2. Setting the value of the anon option to -1 disables anonymous access. Note that, by default, secure NFS accepts nonsecure requests as anonymous, and users who want more security can disable this feature by setting anon to a value of -1.</td>
</tr>
<tr>
<td>root=Client[:Client]</td>
<td>Allows root access from the specified clients in the list. Putting a host in the root list does not override the semantics of the other options. For example, this option denies the mount access from a host present in the root list but absent in the access list.</td>
</tr>
<tr>
<td>secure</td>
<td>Requires clients to use a more secure protocol when accessing the directory. A # (pound sign) anywhere in the file indicates a comment that extends to the end of the line.</td>
</tr>
<tr>
<td>deleg={yes</td>
<td>no}</td>
</tr>
<tr>
<td>vers=version[:version]</td>
<td>Exports the directory for clients using the specified nfs protocol versions. Allowable values are 2, 3, and 4. Versions 2 and 3 cannot be enforced separately. Specifying version 2 or 3 allows access by clients using either nfs protocol versions 2 or 3. Version 4 can be specified independently and must be specified to allow access by clients using version 4 protocol. The default is 2 and 3.</td>
</tr>
<tr>
<td>exname=external-name</td>
<td>Exports the directory by the specified external name. The external name must begin with the <code>nfsroot</code> name. See below for a description of the <code>nfsroot</code> and <code>nfspublic</code> paths. This applies only to directories exported for access by version 4 protocol only.</td>
</tr>
</tbody>
</table>
sec=flavor[:flavor...]

This option is used to specify a list of security methods that may be used to access files under the exported directory. Most `exportfs` options can be clustered using the `sec` option. Options following a `sec` option are presumed to belong with the preceding `sec` option. Any number of `sec` stanzas may be specified, but each security method can be specified only once. Within each `sec` stanza the `ro`, `rw`, `root`, and `access` options may be specified once. Only the public, `anon` and `vers` options are considered global for the export. If the `sec` option is used to specify any security method, it must be used to specify all security methods. In the absence of any `sec` option, `unix` authentication is assumed.

Allowable flavor values are:

- `sys` Unix authentication.
- `dh` DES authentication.
- `krb5` Kerberos. Authentication only.
- `krb5i` Kerberos. Authentication and integrity.
- `krb5p` Kerberos. Authentication, integrity, and privacy.
- `none` Allow mount requests to proceed with anonymous credentials if the mount request uses an authentication flavor not specified in the export. Otherwise a weak `auth` error is returned. By default, all flavors are allowed.

The `secure` option may be specified, but not in conjunction with a `sec` option. The `secure` option is deprecated and may be eliminated. Use `sec=dh` instead.

refer=rootpath@host [+]host[+rootpath@host [+]host]]

A namespace referral is created at the specified path. This referral directs clients to the specified alternate locations where the clients can continue operations. A referral is a special object. If a non-referral object exists at the specified path, the export is not allowed and an error message is printed. If nothing exists at the specified path, a referral object is created there; this referral object includes the pathname directories that lead to the object. Multiple referrals can be created within a filesystem. A referral cannot be specified for `nfsroot`. The name `localhost` cannot be used as a hostname.

Unexporting the referral object has the effect of removing the referral locations information from the referral object. Unexporting the referral object does not remove the referral object itself. The object can be removed using `rm` if desired. The administrator must ensure that appropriate data is available at the referral servers.

This option is available only on AIX version 5.3.0.30 or later, and is allowed only for version 4 exports. If the export specification allows version 2 or version 3 access, an error message will be printed and the export will be disallowed.

**Note:** A referral export can only be made if replication is enabled on the server. Use `chnfs -R on` to enable replication.
Replica location information is associated with the export path. The replica information can be used by NFS version 4 clients to redirect operations to the specified alternate locations if the current server becomes unavailable. You should ensure that appropriate data is available at the replica servers. Since replica information applies to an entire filesystem, the specified path must be the root of a filesystem. If the path is not a filesystem root, the export is not allowed and an error message is printed. The name localhost cannot be used as a hostname.

If the directory being exported is not in the replica list, the entry ExportedDirectory@CurrentHost is added as the first replica location. A replica export can only be made if replication is enabled on the server. By default, replication is not enabled. If replica exports are made at system boot, replication should be enabled using chnfs -R on. Replica locations can also be specified for the nfsroot. The chnfs command must be used for this purpose. In this case, the command is chnfs -R host [ + host ]. If the current host is not specified in the list, it will be added as the first replica host. The rootpath is not needed or allowed in this case. The reason is that the nfsroot is replicated only to the nfsroots of the specified hosts. The replication mode can only be changed if there are no active NFS version 4 exports. If the server's replication mode is changed, any filehandles issued by the server during the previous replication mode will not be honored by the server. This can cause application errors on clients with old filehandles. Care must be taken when changing the replication mode of the server. If possible, all client mounts to the server should be unmounted before the server's replication mode is changed. The replica location information associated with the directory can be changed by modifying the replica list and reexporting the directory. The new replica information will replace the old replica information.

NFS clients are expected to refresh replica information on a regular basis. If the server changes the replica information for an export, it may take some time for the client to refresh its replica information. This is not a serious problem if new replica locations are added, since clients with old replica information will still have correct, though possibly incomplete, replica information. Removing replica information can be problematic since it can result in clients having incorrect replica information for some period of time. To aid clients in detecting the new information, exportfs attempts to touch the replicated directory. This will change the timestamps on the directory, which in turn causes the client to refetch the directory's attributes. This operation may not be possible, however, if the replicated filesystem is read-only.

When changing replica information for a directory, you should be aware that there may be a period of time between the changing of the replica information and clients getting the new replica information.

This option is available only on AIX version 5.3.0.30 or later, and is meaningful only for version 4 exports. If the option is used on an export that allows version 2 or version 3 access, the operation is allowed, but the replica information is ignored by the version 2 and version 3 servers.

noauto Accepts the replicas specification as-is. Does not automatically insert the primary hostname as one of the replica locations if it has not been specified.

**nfsroot and nfspublic**

In order to allow the NFS server administrator to hide some detail of the local file system from clients, the nfsroot and nfspublic attributes were added to the NFS version 4 implementation. The nfsroot and nfspublic may be specified independently, but nfspublic must be a subdirectory of nfsroot. When the nfsroot is set, a local directory can be exported so that it appears to the client to be a subdirectory of the nfsroot. Restrictions must be placed on the exported directories in order to avoid problems:

- The nfsroot must not be "/".
- Either all version 4 exports must specify an external name, or none specify an external.
• The external name must start with the **nfsroot** name. For example, if the **nfsroot** has been set to **/export/server**, the directory **/export/server/abc** may be used as an external name, but the directory **/abc** may not be. In this example, the **/tmp** directory may be exported as **/export/server/tmp**, but not as **/xyz**.

• If a directory is exported with an external name, any descendant of that directory which is also exported must maintain the same path between the two directories. For example, if **/a** is exported as **/export/dira**, the directory **/a/b/c/d** can only be exported as **/export/dira/b/c/d**.

• If a directory is exported with an external name, any parent of that directory which is also exported must maintain the same path between the two directories. For example, if **/a/b** is exported as **/export/a/b**, the directory **/a** can only be exported as **/export/a**. This restriction also says that if **/a/b** is exported as **/export/b**, the directory **/a** can not be exported since it does not exist in the path from the root node to export pathname of **/b**.

• The **exportfs** command will only allow the **exname** option when the **-vers=4** options is also present.

Administration of **nfsroot**, **nfspublic**, and replication is performed using the **chnfs** command.

**Examples**

1. To export to **netgroup** clients, enter:
   ```bash
   /usr -access=clients
   ```

2. To export to the world, enter:
   ```bash
   /usr/local
   ```

3. To export to only these systems, enter:
   ```bash
   /usr2 -access=hermes:zip:tutorial
   ```

4. To give root access only to these systems, enter:
   ```bash
   /usr/tps -root=hermes:zip
   ```

5. To convert client root users to guest UID=100, enter:
   ```bash
   /usr/new -anon=100
   ```

6. To export read-only to everyone, enter:
   ```bash
   /usr/bin -ro
   ```

7. To allow several options on one line, enter:
   ```bash
   /usr/stuff -access=zip,anon=-3,ro
   ```

8. To create a referral at **/usr/info** to the **/usr/info** directory on the host **infoserver**, add the following line to **/etc/exports** and then export **/usr/info**:
   ```bash
   /usr/info -vers=4,refer=/usr/info@infoserver
   ```

9. To specify replicas for the directory **/common/info** at hosts **backup1** and **backup2**, add the following line to **/etc/exports** and then export **/common/info**:
   ```bash
   /common/info -vers=4,replicas=/common/info@backup1:/common/info@backup2,<other options>
   ```

**Files**

- **/etc/xtab**: Lists currently exported directories.
- **/etc/hosts**: Contains an entry for each host on the network.
- **/etc/netgroup**: Contains information about each user group on the network.

**Related Information**

The **chnfs** and **exportfs** commands.

The **nfsd** daemon.

[List of NFS files]
NFS Services in Networks and communication management.

.fig File

Purpose
Contains a list of F file names.

Description
The .fig file is one of several intermediate files produced for each document by InfoCrafter. The .fig file is an ASCII file that contains a list of F file names created for the document. F files are files containing artwork.

Files

.fig Contains a list of F file names.

Related Information
The .srf file.

filesystems File

Purpose
Centralizes file system characteristics.

Description
A file system is a complete directory structure, including a root ( / ) directory and any directories and files beneath it. A file system is confined to a logical volume. All of the information about the file system is centralized in the /etc/filesystems file. Most of the file system maintenance commands take their defaults from this file. The file is organized into stanza names that are file system names and contents that are attribute-value pairs specifying characteristics of the file system.

The filesystems file serves two purposes:
• It documents the layout characteristics of the file systems.
• It frees the person who sets up the file system from having to enter and remember items such as the device where the file system resides, because this information is defined in the file.

File System Attributes
Each stanza names the directory where the file system is normally mounted. The file system attributes specify all the parameters of the file system. The attributes currently used are:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>account</td>
<td>Used by the dodisk command to determine the file systems to be processed by the accounting system. This value can be either the True or False value.</td>
</tr>
<tr>
<td>boot</td>
<td>Used by the mkfs command to initialize the boot block of a new file system. This specifies the name of the load module to be placed into the first block of the file system.</td>
</tr>
<tr>
<td>check</td>
<td>Used by the fsck command to determine the default file systems to be checked. The True value enables checking while the False value disables checking. If a number, rather than the True value is specified, the file system is checked in the specified pass of checking. Multiple pass checking, described in the fsck command, permits file systems on different drives to be checked in parallel.</td>
</tr>
</tbody>
</table>
### Attribute Description

dev
Identifies, for local mounts, either the block special file where the file system resides or the file or directory to be mounted. System management utilities use this attribute to map file system names to the corresponding device names. For remote mounts, it identifies the file or directory to be mounted.

free
This value can be either true or false.
Obsolete and ignored.

mount
Used by the `mount` command to determine whether this file system should be mounted by default. The possible values of the `mount` attribute are:

- **automatic**
  Automatically mounts a file system when the system is started. Unlike the `true` value, filesystems which are mounted with the `automatic` value are not mounted with the `mount all` command or unmounted with the `unmount all` command. By default, the `/`, `/usr`, `/var`, and `/tmp` filesystems use the `automatic` value.

- **false**
  This file system is not mounted by default.

- **true**
  This file system is mounted by the `mount all` command. It is unmounted by the `unmount all` command. The `mount all` command is issued during system initialization to mount automatically all such file systems.

nodename
Used by the `mount` command to determine which node contains the remote file system. If this attribute is not present, the mount is a local mount. The value of the `nodename` attribute should be a valid node nickname. This value can be overridden with the `mount -n` command.

options
Comma-separated list of keywords that have meaning specific to a file system type. The options are passed to the file system at mount time.

size
Used by the `mkfs` command for reference and to build the file system. The value is the number of 512-byte blocks in the file system.

type
Used to group related mounts. When the `mount -t String` command is issued, all of the currently unmounted file systems with a `type` attribute equal to the `String` parameter are mounted.

vfs
Specifies the type of mount. For example, `vfs=nfs` specifies the virtual file system being mounted is an NFS file system.

vol
Used by the `mkfs` command when initializing the label on a new file system. The value is a volume or pack label using a maximum of 6 characters.

log
The `LVName` must be the full path name of the filesystem logging logical volume name to which log data is written as this file system is modified. This is only valid for journaled file systems.

### Examples

The following is an example of a typical `/etc/filesystems` file:

Note: Modifying this file can cause several effects to file systems.

```plaintext
*  
*  File system information
*  
default:
  vol = "root"
  mount = false
  check = false

/: 
  dev = /dev/hd4
  vol = "root"
  mount = automatic
  check = true
  log = /dev/hd8

/home:
  dev = /dev/hd1
  vol = "u"
  mount = true
  check = true
  log = /dev/hd8
```

86   AIX 5L Version 5.3 Files Reference
/home/joe/1:
  dev = /home/joe/1
  nodename = vance
  vfs = nfs

/usr:
  dev = /dev/hd2
  vol = "usr"
  mount = true
  check = true
  log = /dev/hd8

/tmp:
  dev = /dev/hd3
  vol = "tmp"
  mount = true
  check = true
  log = /dev/hd8

Note: The asterisk (*) is the comment character used in the /etc/filesystems file.

Files
/etc/filesystems  Lists the known file systems and defines their characteristics.
/etc/vfs          Contains descriptions of virtual file system types.

Related Information
The backup command, df command, dodisk command, fsck command, mkfs command, mount command, restore command, umount command.

The filesys.h file.
Directories and Logical volume storage in Operating system and device management.

Foreign File for BNU

Purpose
Logs contact attempts from unknown systems.

Description
The /var/spool/uucp/Drum/Foreign file lists access attempts by unknown systems. The /usr/sbin/uucp/remote.unknown shell script appends an entry to the Foreign file each time a remote computer that is not listed in the local /etc/uucp/Systems file attempts to communicate with that local system.

Someone with root user authority can customize entries in the Foreign file to fit the needs of a specific site by modifying the remote.unknown shell script.

Examples
This is a sample entry in the Foreign file:
Wed Sep 20 20:38:22 CDT 1989: call from the system merlin

System merlin, which is not listed in the /etc/uucp/Systems file, attempted to log in September 20 at 20:38 hours (10:38 p.m.). BNU did not allow the unknown system to log in.
Files

/var/spool/uucp/Foreign 
Specifies the path of the Foreign file.

/etc/uucp/Permissions
Describes access permissions for remote systems.

/etc/uucp/Systems
Describes accessible remote systems.

/usr/sbin/uucp/remote.unknown
Records contacts from unknown systems in the Foreign file.

/var/spool/uucp directory
Contains BNU administrative files.

Related Information
The uucp command, uudemon.cleanu command, uux command.
The cron daemon, uucico daemon, uuxqt daemon.

BNU File and Directory Structure and BNU maintenance in Networks and communication management.

.forward File

Purpose
Automatically forwards mail as it is received.

Description
When mail is sent to a local user, the sendmail command checks for the $HOME/.forward file. The $HOME/.forward file can contain one or more addresses or aliases. If the file exists, the message is not sent to the user. The message is sent to the addresses or aliases in the .forward file. For example, if user mickey's .forward file on host disney contains:

```
donald@wonderful.world.disney
pluto
```

Copies of messages sent to mickey are forwarded to user donald on host wonderful.world.disney, and to pluto on the local system.

Notes:
1. The addresses listed in the .forward file can be a comma-separated list of addresses; for example:
   
   ```
donald@wonderful.world.disney, pluto
```

2. Addresses can specify programs. The following example forwards a message to the vacation command:

   ```
mickey, "|/usr/bin/vacation mickey"
```

   This example sends a message to user mickey and to the vacation program.

3. This file must be created by the user in the $HOME directory.

To stop forwarding mail, use the rm command to remove the .forward file from your home directory:

```
rm .forward
```

The .forward file is deleted. Incoming mail is delivered to the user's system mailbox.

Files

$HOME/.forward 
Specifies the path of the file.
Related Information

The `mail` command, `vacation` command. 

Aliases and distribution lists | Forwarding mail | Sending a vacation message notice | Mail program customization options in *Networks and communication management.*

---

### ftpaccess.ctl File

#### Purpose

Specifies FTP host access parameters.

#### Description

The `/etc/ftpaccess.ctl` file is searched for lines that start with `allow:`; `deny:`; `readonly:`; `writeonly:`; `readwrite:`; `useronly:`; `grouponly:`; `herald:` and/or `motd:`. Other lines are ignored. If the file doesn't exist, then ftp access is allowed for all hosts. The `allow:` and `deny:` lines are for restricting host access. The `readonly:`; `writeonly:` and `readwrite:` lines are for restricting ftp reads (get) and writes (put). The `useronly:` and `grouponly:` lines are for defining anonymous users. The `herald:` and `motd:` lines are for multiline messages before and after login.

#### Syntax

The syntax for all lines in `/etc/ftpaccess.ctl` are in the form:

```
keyword: value, value, ...
```

where one can specify one or more values for every keyword. One can have multiple lines with the same keyword. The lines in `/etc/ftpaccess.ctl` are limited to 1024 characters and anything greater than 1024 characters will be ignored. The syntax for the `allow:` and `deny:` lines are:

- `allow:` line allows only the hosts listed in all the `allow:` lines. All other hosts will be refused ftp access. If there are no `allow:` line(s), then all hosts will be given ftp access except those hosts specified in the `deny:` line(s). The host can be specified as either a hostname or IP address.

- `deny:` line denies ftp access for all hosts except those specified in the `allow:` line(s).

The syntax for the `readonly:`; `writeonly:` and `readwrite:` lines are:

- `readonly:` list the readonly directories and the `writeonly:` list the writeonly directories. If one wants read access in a writeonly directory or if one wants write access in a readonly directory, then access is denied. All other directories are granted access except when a `readwrite:` line(s) is specified. If a `readwrite:` line(s) is specified, only directories listed in the `readwrite:` line and/or listed in the `readonly:` line are granted access for reading, and only directories listed in the `readwrite:` line and/or listed in the `writeonly:` line are granted access for writing. Also, these lines can have a value of `ALL` or `NONE`.

The syntax for the `useronly:` and `grouponly:` lines are:

- `useronly:` list the anonymous users and the `grouponly:` list the group of anonymous users. These anonymous users are similar to the user anonymous in that ftp activity is restricted to their home directories.

The `herald:` and `motd:` lines are:

- `herald:` and `motd:` lines are for multiline messages before and after login.
The path is the full path name of the file that contains the multiline herald that will be displayed before login. When the `motd:` line has a value of `ON`, then the `$HOME/motd` file contains the multiline message that will displayed after login. If the user is a defined anonymous user, then the `/etc/motd` file contains the multiline message that will displayed after login. (Note that `/etc/motd` is in the anonymous user’s chroot'ed home directory). The default for the `motd:` line is `OFF`.

### /etc/group File

**Purpose**

Contains basic group attributes.

**Description**

The `/etc/group` file contains basic group attributes. This is an ASCII file that contains records for system groups. Each record appears on a single line and is the following format:

```
Name:Password:ID:User1,User2,...,Usern
```

You must separate each attribute with a colon. Records are separated by new-line characters. The attributes in a record have the following values:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Specifies a group name that is unique on the system.</td>
</tr>
<tr>
<td>Password</td>
<td>Not used. Group administrators are provided instead of group passwords.</td>
</tr>
<tr>
<td>ID</td>
<td>Specifies the group ID. The value is a unique decimal integer string.</td>
</tr>
<tr>
<td>User1,User2,...,Usern</td>
<td>Identifies a list of one or more users. Separate group member names with commas. Each user must already be defined in the local database configuration files.</td>
</tr>
</tbody>
</table>

Do not use a : (colon) in any of the attribute fields. For an example of a record, see the `Examples` section. Additional attributes are defined in the `/etc/security/group` file.

**Note:** Certain system-defined group and user names are required for proper installation and update of the system software. Exercise care before replacing the `/etc/group` file to ensure that no system-supplied groups or users are removed.

You should access the `/etc/group` file through the system commands and subroutines defined for this purpose. You can use the following commands to manage groups:

- `chgroup`
- `chgrpmem`
- `chuser`
- `lsgroup`
- `mkgroup`
- `mkuser`
- `rmgroup`

To change the `Name` parameter, you first use the `mkgroup` command to add a new entry. Then, you use the `rmgroup` command to remove the old group. To display all the attributes in the file, use the `lsgroup` command.
You can use the chgroup, chgrpmem, or chuser command to change all user and group attributes. The mkuser command adds a user whose primary group is defined in the /usr/lib/security/mkuser.default file and the rmuser command removes a user. Although you can change the group ID with the chgroup command, this is not recommended.

**Security**

Access Control: This file should grant read (r) access to all users and grant write (w) access only to the root user and members of the security group.

**Examples**

A typical record looks like the following example for the staff group:

```
staff:!1:shadow,cjf
```

In this example, the GroupID parameter is 1 and the users are defined to be shadow and cjf.

**Files**

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/etc/group</td>
<td>Contains basic group attributes.</td>
</tr>
<tr>
<td>/etc/security/group</td>
<td>Contains the extended attributes of groups.</td>
</tr>
<tr>
<td>/etc/passwd</td>
<td>Contains the basic attributes of users.</td>
</tr>
<tr>
<td>/etc/security/passwd</td>
<td>Contains password information.</td>
</tr>
<tr>
<td>/etc/security/ user</td>
<td>Contains the extended attributes of users.</td>
</tr>
<tr>
<td>/etc/security/ environ</td>
<td>Contains the environment attributes of users.</td>
</tr>
<tr>
<td>/etc/security/ limits</td>
<td>Contains the process resource limits of users.</td>
</tr>
<tr>
<td>/etc/security/audit/config</td>
<td>Contains audit system configuration information.</td>
</tr>
</tbody>
</table>

**Related Information**

The chgroup command, chgrpmem command, lsgroup command, mkgroup command, rmgroup command, setgroups command, setserv command.

The enduserdb subroutine, getgroupattr subroutine, IDto group subroutine, nextgroup subroutine, putgroupattr subroutine, setuserdb subroutine.

**Purpose**

Contains extended group attributes.

**Description**

The /etc/security/group file contains extended group attributes. This is an ASCII file that contains a stanza for each system group. Each stanza is identified by a group name from the /etc/group file followed by a : (colon) and contains attributes in the form Attribute=Value. Each attribute pair ends with a new-line character as does each stanza. The file supports a default stanza. If an attribute is not defined for a group, the default value for the attribute is used.
A stanza can contain one or more of the following attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>adms</td>
<td>Defines the group administrators. Administrators are users who can perform administrative tasks for the group, such as setting the members and administrators of the group. This attribute is ignored if <code>admin = true</code>, since only the root user can alter a group defined as administrative. The value is a list of comma-separated user login-names. The default value is an empty string.</td>
</tr>
<tr>
<td>admin</td>
<td>Defines the administrative status of the group. Possible values are:</td>
</tr>
<tr>
<td></td>
<td>true Defines the group as administrative. Only the root user can change the attributes of groups defined as administrative.</td>
</tr>
<tr>
<td></td>
<td>false Defines a standard group. The attributes of these groups can be changed by the root user or a member of the security group. This is the default value.</td>
</tr>
<tr>
<td>dce_export</td>
<td>Allows the DCE registry to overwrite the local group information with the DCE group information during a DCE export operation. Possible values are:</td>
</tr>
<tr>
<td></td>
<td>true Local group information will be overwritten.</td>
</tr>
<tr>
<td></td>
<td>false Local group information will not be overwritten.</td>
</tr>
<tr>
<td>projects</td>
<td>Defines the list of projects that the user’s processes can be assigned to. The value is a list of comma-separated project names and is evaluated from left to right. The project name should be a valid project name as defined in the system. If an invalid project name is found on the list, it will be reported as an error by the group commands.</td>
</tr>
</tbody>
</table>

For a typical stanza, see the "Examples" section.

You should access the `/etc/security/group` file through the system commands and subroutines defined for this purpose. You can use the following commands to manage groups:

- `mkgroup`
- `chgroup`
- `chgrpmem`
- `lsgrp`
- `rmgroup`

The `mkgroup` command adds new groups to the `/etc/group` file and the `/etc/security/group` file. Use this command to create an administrative group. You can also use the `mkgroup` to set the group administrator.

Use the `chgroup` command to change all the attributes. If you are an administrator of a standard group, you can change the `adms` attribute for that group with the `chgrpmem` command.

The `lsgrp` command displays both the `adms` and the `admin` attributes. The `rmgroup` command removes the entry from both the `/etc/group` file and the `/etc/security/group` file.

To write programs that affect attributes in the `/etc/security/group` file, use the subroutines listed in Related Information.

**Security**

Access Control: This file should grant read (r) access to the root user and members of the security group, and to others as permitted by the security policy for the system. Only the root user should have write (w) access.

Auditing Events:

<table>
<thead>
<tr>
<th>Event</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>S_GROUP_WRITE</td>
<td>file name</td>
</tr>
</tbody>
</table>
Examples
A typical stanza looks like the following example for the finance group:

finance:
   admin = false
   adms = cjf, scott, sah

Files

/etc/security/group
   Specifies the path to the file.
   Contains the basic attributes of groups.
   /etc/group
   Contains the basic attributes of users.
   /etc/passwd
   Contains password information.
   /etc/security/environ
   Contains the extended attributes of users.
   /etc/security/passwd
   Contains the environment attributes of users.
   /etc/security/user
   Contains the process resource limits of users.
   /etc/security/audit/config
   Contains audit system configuration information.
   /etc/security/limits
   Contains last login information.

Related Information
The chgroup command, chgrpmem command, Isgroup command, mkgroup command, rmgrou p command, setgroups command.
The enduserdb subroutine, getgroupattr subroutine, mDigroup subroutine, nextgroup subroutine, putgroupattr subroutine, setuserdb subroutine.

File and system security in Operating system and device management.

Workload Manager groupings File

Purpose
Defines attribute value groupings along with their associated values.

Description
The attribute value groupings file is in the configuration directory. It resides along with the rules file in the SuperConf and SubConf directories.

The attribute value groupings file is formatted as a flat ASCII file list with attribute grouping names followed by an equal (=) sign and the list of all attribute values in the group, separated by commas. The list of attribute values will be terminated by a carriage return. The list of attribute values can be continued onto multiple lines by preceding carriage returns with a backslash. The only whitespace that is significant in the groupings file is a carriage return. Other whitespace characters are removed during file parsing. Comments are lines preceded by an asterisk.

Each attribute grouping definition is limited to WLM_GROUPING_LEN characters. The attribute grouping name and the list of attribute values cannot be an empty string.

Use of Attribute Groupings
Attribute groupings can be used as element of a selection criteria in the rules file for superclasses or subclasses. The attribute grouping name must be preceded by a dollar sign ($) and will be replaced by the list of all attribute values associated with itself. No special character (*,[,-,?) except exclusion character '!' can be applied to an attribute grouping name. Attribute groupings cannot been used in the class field.
"rules" files:
* class resvd user group application type tag
classA - $trusted,!$nottrusted - -
classB - - - $shell,$/bin/zsh - -
classC - - $rootgroup -

Syntax
The syntax of the attribute values is the same as in the [rules] file, including potential wildcards ([,]*,-,?,+). The use of the exclusion character '!' in the attribute values list is not allowed. This restriction is necessary to avoid a confusing interpretation of an attribute value grouping used in the class assignment file preceded by an exclusion character. Syntax is checked only when attribute groupings are used (rules processing during a configuration load or explicit check with wlmcheck command). The groupings file is not mandatory. By default, no attribute grouping is defined. Attribute value groupings of a groupings file are defined and usable only in the scope of their configuration directory (SuperConfDir or SubConfDir level). If it exists, the groupings file is copied in the .running directory when the configuration is loaded into the kernel as it is done with other configuration files. No command interface is provided to update the attribute groupings file.

Example
"groupings" file:
* attribute groupings definition
* will be used in the rules file
trusted = user[0-9][0-9],admin*
nottrusted = user23, user45
shell=/bin/?sh,\n  /bin/sh,\n  /bin/tcsh
rootgroup=system,bin,sys,security,cron,audit

Files
$HOME/groupings Defines attribute value groupings along with their associated values.

Related Information
The rules file.

hostmibd.conf File

Purpose
Defines the configuration parameters for hostmibd dpi2 sub-agent.

Description
The hostmibd.conf file provides the configuration information for the hostmibd dpi2 sub-agent. This file can be changed while the hostmibd dpi2 sub-agent is running. If the refresh command is issued, the hostmibd dpi2 sub-agent will reread this configuration file. The hostmibd dpi2 sub-agent must be under System Resource Control (SRC) for the refresh command to force the reread. To accomplish the reread, as root user, run:

# refresh -s hostmibd

Keywords
The directives are specified in the form of <keyword>=<value>. The keyword is case-insensitive. The value passed is also case-sensitive.
LogFilename
The name of the most recent log file. Less recent log files have the number 1 to (n - 1) appended to their names. The larger the number, the less recent the file.

logFileSize
The size of log files in K bytes. Maximum size of a log file. When the size of the most recent log file reaches this value, it is renamed and a new log file is created.

numLogFiles
The number of log files desired. The maximum value for numLogFiles is 4. A new file is created when the size of the log file is equal or more than the size specified by the keyword logFileSize. When the number of log files reaches the numLogFiles the log files start rotating.

requestTimeout
The timeout in seconds that the snmpd agent will wait for a response from this sub-agent. The default value is 60 seconds.

tracelevel
The tracing/debug level to do.

0 = Least level
8 = DPI level 1
16 = DPI level 2
32 = Internal level 1
64 = Internal level 2
128 = Internal level 3

Add the numbers for multiple trace levels.

updateInterval
The interval, in seconds, that the sub-agent will use to refresh its internal table. The default value is 30 seconds.

Example
logFileName=/usr/tmp/hostmibd.log
logFileSize=0
numLogFiles=0
requestTimeout=180
tracelevel=0
updateInterval=120

Files
/etc/hostmibd.conf Defines the configuration parameters for hostmibd dpi2 sub-agent.

Related Information
The hostmibd, snmpd, and refresh commands.

image.data File

Purpose
Contains information on the image installed during the Base Operating System installation process.

Description
The image.data file contains information describing the image installed during the BOS installation process. This information includes the sizes, names, maps, and mount points of logical volumes and file systems in the root volume group. The mkszfile command generates the image.data file. It is not
recommended that the user modify the file. Changing the value of one field without correctly modifying any
related fields can result in a failed installation and a corrupted backup image. The only exception to this
recommendation is the SHRINK field, which the user may modify to instruct the BOS installation routines to
create the file systems as specified in the image.data file or to create the file systems only as large as is
required to contain all the data in the file system.

The BOS installation process also takes input from the image.data file regarding defaults for the machine
being installed. Any default values in the image.data file will override values obtained when the BOS
installation queries the hardware topology and existing root volume group. The image.data file resides in
the / directory.

This file is part of System Backup and BOS Install Utilities.

The image.data file is arranged in stanza format. Each stanza contains one or more fields. These stanzas
include the following:

- **image_data**
- **logical_volume_policy**
- **ils_data**
- **vg_data**
- **source_disk_data**
- **lv_data**
- **fs_data**
- **post_install_data**
- **post_restvg**

**image_data Stanza**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMAGE_TYPE</td>
<td>Identifies the format of the image. Examples include backup file format (bff) and tar format.</td>
</tr>
<tr>
<td>DATE_TIME</td>
<td>Contains the date and time that the image was taken.</td>
</tr>
<tr>
<td>UNAME_INFO</td>
<td>Identifies the system and system level data associated with the image.</td>
</tr>
<tr>
<td>PRODUCT_TAPE</td>
<td>Specifies whether the image is a product image or a mksysb image. The possible field values are yes or no.</td>
</tr>
<tr>
<td>USERVG_LIST</td>
<td>Lists the user volume groups defined in the system.</td>
</tr>
<tr>
<td>OSLEVEL</td>
<td>Identifies the version.release.maintenance.fix level of the system at the time the image was taken</td>
</tr>
</tbody>
</table>

**Note:** The PRODUCT_TAPE and USERVG_LIST fields are only present for the ROOTVG volume group.

**logical_volume_policy Stanza**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHRINK</td>
<td>Instructs BOS install routines to create the file systems as they are specified in the image.data file or create the smallest file systems required to contain all the data in the file system. The field value specified can be yes (shrink file systems) or no (use image.data file specifications).</td>
</tr>
<tr>
<td>EXACT_FIT</td>
<td>The field value specified can be yes or no. If yes is specified, the disk information listed in the source_disk_data stanza must match the actual disks found on the target machine during installation.</td>
</tr>
</tbody>
</table>
**ils_data Stanza**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LANG</td>
<td>Sets the language used by the BOS Install program.</td>
</tr>
</tbody>
</table>

**vg_data Stanza**

**Notes:**
1. The image.data file can contain only one vg_data stanza.
2. Starting with AIX 4.3.3, two new fields (BIGVG and TFACTOR) have been added to the vg_data Stanza

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGNAME</td>
<td>Specifies the volume group name.</td>
</tr>
<tr>
<td>PPSIZE</td>
<td>Specifies the size of the physical partition for the volume group.</td>
</tr>
<tr>
<td>VARYON</td>
<td>Activates the volume group and all associated logical volumes so that the volume group is available for use. The field value can be yes or no.</td>
</tr>
<tr>
<td>VG_SOURCE_DISK_LIST</td>
<td>Lists the disks in the volume group.</td>
</tr>
<tr>
<td>QUORUM</td>
<td>If set to 1, indicates the volume group is to be automatically varied off after losing its quorum of physical volumes.</td>
</tr>
<tr>
<td>CONC_AUTO</td>
<td>Indicates a volume group is to be varied on automatically in concurrent mode.</td>
</tr>
<tr>
<td>BIGVG</td>
<td>Indicates a volume group is to be created as a big vg format volume group. This can accommodate up to 128 physical volumes and 512 logical volumes.</td>
</tr>
<tr>
<td>TFACTOR</td>
<td>Indicates a change in the limit of the number of physical partitions per physical volume.</td>
</tr>
<tr>
<td>ENH_CONC_CAPABLE</td>
<td>Indicates a volume group is enhanced concurrent capable.</td>
</tr>
</tbody>
</table>

**source_disk_data Stanza**

**Note:** The image.data file contains one source_disk_data stanza for each disk in the root volume group.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PVID</td>
<td>Specifies the 16 digit physical volume identifier for the disk.</td>
</tr>
<tr>
<td>CONNECTION</td>
<td>Specifies the combination of the parent and the conwhere attribute associated with a disk. The format for this field is: parent attribute/conwhere attribute.</td>
</tr>
<tr>
<td>LOCATION</td>
<td>Specifies the locations of the disks in the root volume group.</td>
</tr>
<tr>
<td>SIZE_MB</td>
<td>Specifies the size, in MB, of the disks in the root volume group.</td>
</tr>
<tr>
<td>HDISKNAME</td>
<td>Specifies the names of the disks in the root volume group.</td>
</tr>
</tbody>
</table>

**lv_data Stanza**

**Note:** The image.data file contains one lv_data stanza for each logical volume created on the system.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VOLUME_GROUP</td>
<td>Specifies the logical volume group name. Volume group names must be unique, system wide, and can range from 1 to 15 characters.</td>
</tr>
<tr>
<td>LV_SOURCE_DISK_LIST</td>
<td>Lists the disks in the logical volume.</td>
</tr>
<tr>
<td>LV_IDENTIFIER</td>
<td>Contains the identifier of the logical volume.</td>
</tr>
<tr>
<td>LOGICAL_VOLUME</td>
<td>Contains the name of the logical volume.</td>
</tr>
<tr>
<td>PERMISSION</td>
<td>Sets the access permissions. The field value can be read/write or read only.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>VG_STAT</td>
<td>Indicates the state of the volume group. If the volume group is activated with the <code>varyonvg</code> command, the value of the VG_STAT field is either active/complete or active/partial. An active/complete field value indicates that all physical volumes are active, while an active/partial field value indicates that all physical volumes are not active. If the volume group is not activated with the <code>varonvg</code> command, the VG_STAT field value is inactive.</td>
</tr>
<tr>
<td>TYPE</td>
<td>Describes the logical volume type.</td>
</tr>
<tr>
<td>MAX_LPS</td>
<td>Sets the maximum number of logical partitions within the logical volume.</td>
</tr>
<tr>
<td>COPIES</td>
<td>Specifies the number of physical partitions created for each logical partition during the allocation process.</td>
</tr>
<tr>
<td>LPS</td>
<td>Specifies the number of logical partitions currently in the logical volume.</td>
</tr>
<tr>
<td>STALE_PPs</td>
<td>Specifies the number of physical partitions in the logical volume that are not current.</td>
</tr>
<tr>
<td>INTER_POLICY</td>
<td>Specifies the inter-physical allocation policy. The field value can be minimum or maximum.</td>
</tr>
<tr>
<td>INTRA_POLICY</td>
<td>Specifies the intra-physical allocation policy. The possible field values are either middle, center, or edge.</td>
</tr>
<tr>
<td>MOUNT_POINT</td>
<td>Specifies the file-system mount point for the logical volume, if applicable.</td>
</tr>
<tr>
<td>MIRROR_WRITE_CONSISTENCY</td>
<td>Specifies mirror-write consistency state. The field value can be off or on.</td>
</tr>
<tr>
<td>LV_SEPARATE_PV</td>
<td>Specifies a yes, no, or super field value for strict allocation. A yes value for strict allocation states that no copies for a logical partition are allocated on the same physical volume. A no value for strict allocation (non-strict) states that at least one occurrence of two physical partitions belong to the same logical partition. A super value for strict allocation (super strictness) states that no partition from one mirror copy may reside on the same disk as another mirror copy.</td>
</tr>
<tr>
<td>LV_STATE</td>
<td>Describes the state of the logical volume. An Opened/stale value indicates the logical volume is open but contains physical partitions that are not current. An Open/syncd value indicates the logical volume is open and its physical partitions are current, or synchronized. A Closed value indicates the logical volume has not been opened.</td>
</tr>
<tr>
<td>WRITE_VERIFY</td>
<td>Specifies the field value of the write verify state as on or off.</td>
</tr>
<tr>
<td>PP_SIZE</td>
<td>Provides the size physical partition.</td>
</tr>
<tr>
<td>SCHED_POLICY</td>
<td>Specifies a sequential or parallel scheduling policy.</td>
</tr>
<tr>
<td>PP</td>
<td>Specifies the number of physical partitions currently in the logical volume.</td>
</tr>
<tr>
<td>BB_POLICY</td>
<td>Specifies the bad block relocation policy.</td>
</tr>
<tr>
<td>RELOCATABLE</td>
<td>Indicates whether the partitions can be relocated if a reorganization of partition allocation takes place. The field value can be yes or no.</td>
</tr>
<tr>
<td>UPPER_BOUND</td>
<td>Specifies the maximum number of physical volumes for allocation.</td>
</tr>
<tr>
<td>LABEL</td>
<td>Specifies the label field for the logical volume.</td>
</tr>
<tr>
<td>MAPFILE</td>
<td>Provides the full path name to a map file to be used in creating the logical volume.</td>
</tr>
<tr>
<td>LV_MIN_LPS</td>
<td>Specifies the number of physical volumes being restricted when shrinking the logical volume.</td>
</tr>
<tr>
<td>STRIPE_WIDTH</td>
<td>Specifies the number of bytes per strip. Strip size multiplied by the number of disks in the array equals the stripe size. The field value must be a power of two, between 4KB and 128MB; for example, 4KB, 8KB, 16KB, 32KB, 64KB, 128KB, 256KB, 512KB, 1MB, 2MB, 4MB, 8MB, 16MB, 32MB, 64MB, 128MB.</td>
</tr>
<tr>
<td>STRIP_SIZE</td>
<td></td>
</tr>
<tr>
<td>SERIALIZE_IO</td>
<td>Turns on/off serialization of overlapping IOs. If serialization is turned on, then overlapping IOs are not allowed on a block range and only a single IO in a block range is processed at any one time. Most applications (file systems and databases) do serialization, so serialization should be turned off.</td>
</tr>
</tbody>
</table>
**fs_data Stanza**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FS_NAME</td>
<td>Specifies the mount point of the file system.</td>
</tr>
<tr>
<td>FS_SIZE</td>
<td>Specifies the size, in 512-byte blocks, of the file system.</td>
</tr>
<tr>
<td>FS_MIN_SIZE</td>
<td>Specifies the minimum size required to contain the files of the file system.</td>
</tr>
<tr>
<td></td>
<td>This size is used when the SHRINK field in the logical_volume_policy stanza has a field value of yes.</td>
</tr>
<tr>
<td>FS_LV</td>
<td>Provides the logical volume name. The name must contain the /dev/ prefix.</td>
</tr>
<tr>
<td></td>
<td>An example of an appropriate name is /dev/hd4.</td>
</tr>
<tr>
<td>FS_FS</td>
<td>Specifies the fragmentation size of the system. This value is optional.</td>
</tr>
<tr>
<td>FS_NBPI</td>
<td>Specifies the number of bytes per inode. This value is optional.</td>
</tr>
<tr>
<td>FS_COMPRESS</td>
<td>Designates whether the file system should be compressed or not. The field value can be LZ, which compresses the file system, or the no field value.</td>
</tr>
<tr>
<td>FS_BF</td>
<td>Enables the file system for files greater than 2 GB. The possible values are true or false.</td>
</tr>
<tr>
<td>FS_AGSIZE</td>
<td>Specifies the allocation group size. The possible values are 8, 16, 32, or 64. The allocation group size is specified in units of megabytes.</td>
</tr>
<tr>
<td>FS_JFS2_BS</td>
<td>Specifies the file system block size in bytes, 512, 1024, 2048, or 4096 bytes.</td>
</tr>
<tr>
<td>FS_JFS2_SPARSE</td>
<td>Specifies when files are created with holes. The enhanced journaled file system (JFS2) allocates disk blocks for those holes and fills the holes with 0s.</td>
</tr>
<tr>
<td>FS_JFS2_INLINELOG</td>
<td>Specifies that the journal log for the enhanced journaled file system (JFS2) is within the file system.</td>
</tr>
<tr>
<td>FS_JFS2_SIZEINLINELOG</td>
<td>Specifies the size, in megabytes, for the optional inline journal log. The default is the size of the enhanced journaled file system (JFS2) divided by 256.</td>
</tr>
</tbody>
</table>

**post_install_data Stanza**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOSINST_FILE</td>
<td>Provides the full path name of a file or command to execute after BOS install completes.</td>
</tr>
</tbody>
</table>

**post_restvg Stanza**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESTVG_FILE</td>
<td>Specifies the full path name of the file or command to execute after the restvg process completes.</td>
</tr>
</tbody>
</table>

**Note:** The post_install_data stanza exists for the ROOTVG volume group and the post_restvg stanza is present for other volume groups.

**Related Information**

The mkszfile command, mkfs command, mkiv command, and lsv command.

---

**/etc/security/.ids File**

**Purpose**

Contains standard and administrative user IDs and group IDs.

**Description**

The /etc/security/.ids file keeps a count so that every UID (userid) and GID (groupid) has its own unique number. It is not recommended that you edit this file unless it is absolutely necessary.
Example
7 201 11 200

The first number in the example (7) will be the User ID of the next administrative user created on the system. The second number (201) will be the User ID of the next regular user created on the system. The third number (11) is the next administrative Group ID (GID) that will be used when an administrative user is created on the system. The fourth number (200) is the next regular user GID used when a user is created.

Location
/etc/security/.ids Location of the .ids file.

Related Information
The [mkusr] command and the [mkgroup] command.

INed Files

Purpose
Contains programs and data used by the INed program.

Description
The /usr/lib/INed directory contains a number of files and subdirectories used internally by the INed program. The /usr/lib/nls/msg/$LANG directory contains files of translatable text. This directory also contains other files that are not used by INed.

In the following file names, $LANG is the value of the lib/Language environment variable, which indicates the national language currently being used.

<table>
<thead>
<tr>
<th>Directory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bin</td>
<td>Directory containing programs called by the editor to perform various functions. Do not run these programs from the command line.</td>
</tr>
<tr>
<td>FATAL.LOG</td>
<td>Log of error messages the editor records when it encounters a system problem.</td>
</tr>
<tr>
<td>helpers</td>
<td>Directory containing programs called by the editor to help work on certain kinds of data. Files ending in .x or named x use the helper named x.help. Helpers typically supply the functions listed on the INed local menus.</td>
</tr>
<tr>
<td>forms</td>
<td>Directory containing forms used by the INed program. Files ending in .x or named x use the x.ofm form. The forms are binary files used directly by the editor in generating displays for structured files.</td>
</tr>
<tr>
<td>/usr/lib/nls/msg/$LANG/keys.map</td>
<td>File displayed when the Help command key (F1) is pressed and the keymap option is selected.</td>
</tr>
<tr>
<td>termcap</td>
<td>Directory containing the files used by the editor to read input from the terminals and write output to the terminals. The def.trm file is the readable structured file, and the terms.bin file is the compressed version.</td>
</tr>
<tr>
<td>/usr/lib/nls/msg/$LANG</td>
<td>Directory containing help message files and other files containing translated text used by the INed editor. This directory also contains other files not used by INed.</td>
</tr>
</tbody>
</table>
**Files**

/usr/lib/INed directory
Contains files and subdirectories used by the INed program.

/usr/lib/nls/msg/SLANG directory
Contains files of translatable text.

**Related Information**
The `at`, `cat`, `format`, `nl`, `piobe`, `qprt`, `sort`, `stty`, `trbsd`, `untab` command.

---

**.info File**

**Purpose**
Stores configuration information used by the Network Install Manager (NIM).

**Description**
The .info file contains a series of Korn shell variable assignments used by NIM. The .info file is created by NIM for each client. During network boot, the rc.boot program uses several of these variables to control processing.

If a client is initialized by NIM, the .info file is copied into that client’s /etc directory as the /etc/niminfo file. The nimclient command uses the /etc/niminfo file to communicate with the NIM master server.

**Note:** The following variable groups are based upon the function of the variables that they contain. The .info file itself is not divided into categories.

**Variables used directly by the rc.boot program**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROUTES</td>
<td>Contains all the routing information the client needs in order to access any allocated NIM resource. This information is presented as a series of space-separated stanzas, each in the following format:</td>
</tr>
<tr>
<td></td>
<td>DestinationIPAddress:DestinationSubnet:GatewayIPAddress</td>
</tr>
<tr>
<td>SPOT</td>
<td>Specifies the location of the shared product object tree (SPOT) to be used during the boot process. This variable contains the host and pathname of the client’s SPOT in the following format:</td>
</tr>
<tr>
<td></td>
<td>HostName:SPOTDirectory</td>
</tr>
<tr>
<td>RC_CONFIG</td>
<td>Specifies the file name of the rc.config script to use.</td>
</tr>
<tr>
<td>NIM_HOSTS</td>
<td>Provides information used to construct an /etc/hosts file for the client. The value is formatted as follows:</td>
</tr>
<tr>
<td></td>
<td>IPAddress:HostName IPAddress:HostName ...</td>
</tr>
</tbody>
</table>

**Variables used by any rc.config script**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROOT</td>
<td>Specifies the host and path name of the client’s root directory in the following format:</td>
</tr>
<tr>
<td></td>
<td>HostName:RootDirectory</td>
</tr>
<tr>
<td>MOUNTS</td>
<td>Contains a series of space-separated stanzas, each composed of a remote directory specification and the point where it should be mounted. The stanzas are in the following format:</td>
</tr>
<tr>
<td></td>
<td>HostName:RemoteDirectory:LocalDirectory</td>
</tr>
</tbody>
</table>
Variables used by the nim commands

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NIM_NAME</td>
<td>Designates the name of the client’s NIM machines object.</td>
</tr>
<tr>
<td>NIM_CONFIGURATION</td>
<td>Specifies the client’s NIM configuration machine type.</td>
</tr>
<tr>
<td>NIM_MASTER</td>
<td>Specifies the IP address of the NIM master server.</td>
</tr>
<tr>
<td>NIM_MASTER_PORT</td>
<td>Specifies the port number to use for client communications.</td>
</tr>
<tr>
<td>NIM_REGISTRATION_PORT</td>
<td>Specifies the port number to use for client registration.</td>
</tr>
<tr>
<td>NIM_MAX_RETRIES</td>
<td>Specifies the maximum number of retries for communication attempts with the nimesis daemon.</td>
</tr>
<tr>
<td>NIM_MAX_DELAY</td>
<td>Sets the amount of time to wait between retries for communication with the nimesis daemon.</td>
</tr>
</tbody>
</table>

Variables used by BOS Install

The following variables are used by NIM to control Base Operating System (BOS) installation operation:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NIM_BOSINST_DATA</td>
<td>Specifies the RAM file system path name to the bosinst.data file to be used. This variable has the following format: path name</td>
</tr>
<tr>
<td>NIM_BOS_IMAGE</td>
<td>Specifies the RAM file system path name to the BOS image.</td>
</tr>
<tr>
<td>NIM_CUSTOM</td>
<td>Specifies the path name of the customization script to execute after BOS installation.</td>
</tr>
</tbody>
</table>

Variables used by the rc.dd_boot Script

The rc.dd script uses the following variables to perform boot specific processing to create certain NIM resources.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTLS_PAGING_SIZE</td>
<td>Contains the paging-space size that you specify. If you have not set the paging space, the value is NULL and the rc.dd_boot script defaults to a paging space twice that of the client’s RAM space.</td>
</tr>
<tr>
<td>DTLS_LOCAL_FS</td>
<td>Contains a list of acronyms specifying the filesystems to be created locally on the client. The possible values are tmp and home.</td>
</tr>
</tbody>
</table>

Examples

The following is an example of a .info file:

```sh
#----------------Network Install Manager--------
# warning - this file contains NIM configuration information
# and should only be updated by NIM
export NIM_NAME=dua
export NIM_CONFIGURATION=standalone
export NIM_MASTER_HOSTNAME=satu
export NIM_MASTER_PORT=1058
export NIM_REGISTRATION_PORT=1059
export RC_CONFIG=rc.bos_inst
export SPOT=tiga:/usr
export NIM_CUSTOM=/tmp/dua.script
export NIM_BOS_IMAGE=SPOT
export NIM_BOS_FORMAT=master
export NIM_HOSTS="130.35.130.1:satu 130.35.130.3:tiga "
export MOUNTS="tiga:/export/logs/dua:/var/adm/ras:dir
tiga:/export/nim/simages
:/SPOT/usr/sys/inst.images:dir
satu:/export/nim/scripts/dua.script:tmp/dua.script:file 
```

102 AIX 5L Version 5.3 Files Reference
Related Information
The \texttt{lsnim} command, \texttt{nim} command, \texttt{nimclient} command, \texttt{nimconfig} command, \texttt{niminit} command.

\textbf{inittab File}

\textbf{Purpose}
Controls the initialization process.

\textbf{Description}
The \texttt{/etc/inittab} file supplies the script to the \texttt{init} command’s role as a general process dispatcher. The process that constitutes the majority of the \texttt{init} command’s process dispatching activities is the \texttt{/etc/getty} line process, which initiates individual terminal lines. Other processes typically dispatched by the \texttt{init} command are daemons and the shell.

The \texttt{/etc/inittab} file is composed of entries that are position-dependent and have the following format:

\texttt{Identifier:RunLevel:Action:Command}

\textbf{Note:} The colon character (:) is used as a delimiter as well as a comment character. To comment out an \texttt{inittab} entry, add \texttt{:} at the beginning of the entry. For example:

\texttt{:Identifier:RunLevel:Action:Command}

Each entry is delimited by a newline character. A backslash (\) preceding a newline character indicates the continuation of an entry. There are no limits (other than maximum entry size) on the number of entries in the \texttt{/etc/inittab} file. The maximum entry size is 1024 characters. The entry fields are:

\textbf{Identifier}
A string (one or more than one character) that uniquely identifies an object.

\textbf{RunLevel}
The run level in which this entry can be processed. Run levels effectively correspond to a configuration of processes in the system. Each process started by the \texttt{init} command is assigned one or more run levels in which it can exist. Run levels are represented by the numbers 0 through 9. For example, if the system is in run level 1, only those entries with a 1 in the \texttt{runlevel} field are started. When you request the \texttt{init} command to change run levels, all processes without an entry in the \texttt{runlevel} field for the target run level receive a warning signal (SIGTERM). There is a 20-second grace period before processes are forcibly terminated by the kill signal (SIGKILL). The \texttt{runlevel} field can define multiple run levels for a process by selecting more than one run level in any combination from 0 through 9. If no run level is specified, the process is assumed to be valid at all run levels.

There are three other values that appear in the \texttt{runlevel} field, even though they are not true run levels: a, b, and c. Entries that have these characters in the \texttt{runlevel} field are processed only when the \texttt{telinit} command requests them to be run (regardless of the current run level of the system). They differ from run levels in that the \texttt{init} command can never enter run level a, b, or c. Also, a request for the execution of any of these processes does not change the current run level. Furthermore, a process started by an a, b, or c command is not killed when the \texttt{init} command changes levels. They are only killed if their line in the \texttt{/etc/inittab} file is marked off in the \texttt{action} field, their line is deleted entirely from \texttt{/etc/inittab}, or the \texttt{init} command goes into single-user mode.

\textbf{Action} Tells the \texttt{init} command how to treat the process specified in the \texttt{process} field. The following actions are recognized by the \texttt{init} command:

\texttt{respawn}
If the process does not exist, start the process. Do not wait for its termination (continue
scanning the `/etc/inittab` file). Restart the process when it dies. If the process exists, do nothing and continue scanning the `/etc/inittab` file.

**wait**  
When the `init` command enters the run level that matches the entry’s run level, start the process and wait for its termination. All subsequent reads of the `/etc/inittab` file while the `init` command is in the same run level will cause the `init` command to ignore this entry.

**once**  
When the `init` command enters a run level that matches the entry’s run level, start the process, and do not wait for its termination. When it dies, do not restart the process. When the system enters a new run level, and the process is still running from a previous run level change, the program will not be restarted. All subsequent reads of the `/etc/inittab` file while the `init` command is in the same run level will cause the `init` command to ignore this entry.

**boot**  
Process the entry only during system boot, which is when the `init` command reads the `/etc/inittab` file during system startup. Start the process, do not wait for its termination, and when it dies, do not restart the process. In order for the instruction to be meaningful, the run level should be the default or it must match the `init` command’s run level at boot time. This action is useful for an initialization function following a hardware reboot of the system.

**bootwait**  
Process the entry the first time that the `init` command goes from single-user to multi-user state after the system is booted. Start the process, wait for its termination, and when it dies, do not restart the process. If the `initdefault` is 2, run the process right after boot.

**powerfail**  
Execute the process associated with this entry only when the `init` command receives a power fail signal (`SIGPWR`).

**powerwait**  
Execute the process associated with this entry only when the `init` command receives a power fail signal (`SIGTERM`), and wait until it terminates before continuing to process the `/etc/inittab` file.

**off**  
If the process associated with this entry is currently running, send the warning signal (`SIGTERM`), and wait 20 seconds before terminating the process with the kill signal (`SIGKILL`). If the process is not running, ignore this entry.

**ondemand**  
Functionally identical to `respawn`, except this action applies to the a, b, or c values, not to run levels.

**initdefault**  
An entry with this action is only scanned when the `init` command is initially invoked. The `init` command uses this entry, if it exists, to determine which run level to enter initially. It does this by taking the highest run level specified in the `runlevel` field and using that as its initial state. If the `runlevel` field is empty, this is interpreted as 0123456789; therefore, the `init` command enters run level 9. Additionally, if the `init` command does not find an `initdefault` entry in the `/etc/inittab` file, it requests an initial run level from the user at boot time.

**sysinit**  
Entries of this type are executed before the `init` command tries to access the console before login. It is expected that this entry will only be used to initialize devices on which the `init` command might try to ask the run level question. These entries are executed and waited for before continuing.
Command
A shell command to execute. The entire command field is prefixed with exec and passed to a
forked sh as sh -c exec command. Any legal sh syntax can appear in this field. Comments can be
inserted with the # comment syntax.

The getty command writes over the output of any commands that appear before it in the inittab
file. To record the output of these commands to the boot log, pipe their output to the alog -tboot
command.

The stdin, stdout and stdferr file descriptors may not be available while init is processing inittab
entries. Any entries writing to stdout or stderr may not work predictably unless they redirect their
output to a file or to /dev/console.

The following commands are the only supported method for modifying the records in the /etc/inittab file:

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>chitab</td>
<td>Changes records in the /etc/inittab file.</td>
</tr>
<tr>
<td>lsitab</td>
<td>Lists records in the /etc/inittab file.</td>
</tr>
<tr>
<td>mkitab</td>
<td>Adds records to the /etc/inittab file.</td>
</tr>
<tr>
<td>rmitab</td>
<td>Removes records from the /etc/inittab file.</td>
</tr>
</tbody>
</table>

Examples
1. To start the ident process at all run levels, enter:
   ident:0123456789:Action:Command
2. To start the ident process only at run level 2, enter:
   ident:2:Action:Command
3. To disable run levels 0, 3, 6-9 for the ident process, enter:
   ident:1245:Action:Command
4. To start the rc command at run level 2 and send its output to the boot log, enter:
   rc:2:wait:/etc/rc 2>&1 | alog -tboot > /dev/console

Files
/etc/inittab
/usr/sbin/getty

Related Information
The chitab command, init command, lsitab command, mkitab command, rmitab command, telinit
command.

irs.conf File

Purpose
Specifies the ordering of certain name resolution services.

Description
The /etc/irs.conf file is used to control the order of mechanisms that the resolver libraries use in searching
for network-related data. The following subroutines resolve host names, networks, services, protocols, and
netgroups:
Because these subroutines are commonly used in many TCP/IP applications, using the `irs.conf` file can control the directions of the queries made by these applications as well.

By default, the subroutines use the lookup mechanisms to resolve host names in this order:
1. Domain Name Server (DNS)
2. Network Information Service (NIS), if active
3. local files

By default, the subroutines use the lookup mechanisms to resolve networks in this order:
1. DNS
2. NIS, if active
3. local files

By default, the subroutines use the lookup mechanisms to resolve other maps in this order:
1. NIS, if active
2. local files

You can override the default order by modifying the `/etc/irs.conf` configuration file and specifying the desired ordering.

The settings in the `/etc/netsvc.conf` configuration file override the settings in the `/etc/irs.conf` file. The `NSORDER` environment variable overrides the settings in the `/etc/irs.conf` and the `/etc/netsvc.conf` files.

To use DNS to obtain information concerning netgroups, protocols, and services, you must create and use a Hesiod DNS Zone in the following format:

```
map mechanism [option]
```

The following values are available for the `map` parameter:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>services</td>
<td>Lists the port numbers, transport protocols, and names of well-known services</td>
</tr>
<tr>
<td>protocols</td>
<td>Retrieves official names and protocol numbers of protocol aliases</td>
</tr>
<tr>
<td>hosts</td>
<td>Defines the mappings between host names and their IP addresses</td>
</tr>
<tr>
<td>networks</td>
<td>Retrieves network names and addresses</td>
</tr>
<tr>
<td>netgroup</td>
<td>Retrieves groups of hosts, networks, and users in this group</td>
</tr>
</tbody>
</table>

The following values are available for the `mechanism` parameter:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>local</td>
<td>Examines local configuration files (<code>/etc/hosts</code>, <code>/etc/protocols</code>, <code>/etc/services</code>, <code>/etc/netgroup</code>, and <code>/etc/networks</code> files)</td>
</tr>
<tr>
<td>dns</td>
<td>Queries DNS; the <code>/etc/resolv.conf</code> file must be configured for this mechanism to work.</td>
</tr>
<tr>
<td>nis</td>
<td>Queries NIS; the NIS client must be active on the system for this mechanism to work.</td>
</tr>
<tr>
<td>nis+</td>
<td>Queries NIS+; The NIS+ client must be active on the system for this mechanism to work.</td>
</tr>
</tbody>
</table>
Value | Description
---|---
ldap | Queries the LDAP server; the `resolv.ldap` file must be configured for this mechanism to work. **Note:** You can only assign the value `hosts` to the `map` parameter for `ldap`. Although still supported, the use of the `ldap` mechanism is deprecated. Use of the `nis.ldap` mechanism is recommended.
nis.ldap | Queries the LDAP server configured in the `ldap.cfg` file. The LDAP client should be set up on the system using the `mksecldap` command, to use this mechanism. All map types are supported by `nis.ldap`.
local4 | Examines local configuration files for IPv4 addresses.
local6 | Examines local configuration files for IPv6 addresses.
dns4 | Queries DNS for A records (IPv4 addresses); the `/etc/resolv.conf` file must be configured for this mechanism to work.
dns6 | Queries DNS for AAAA records (IPv6 addresses); the `/etc/resolv.conf` file must be configured for this mechanism to work.
nis4 | Queries NIS for information about IPv4 addresses; the NIS client must be active on the system to use this mechanism.
nis6 | Queries NIS for information about IPv6 addresses; the NIS client must be active on the system to use this mechanism.
ldap4 | Queries the LDAP server for information about IPv4 addresses.
ldap6 | Queries the LDAP server for information about IPv6 addresses.

The following values are available for the `option` parameter:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>continue</td>
<td>If the information is not found in the specified mechanism, then instructs the resolver to continue to the next line, which should list another mechanism for the same map</td>
</tr>
<tr>
<td>merge</td>
<td>Merges all responses from multiple <code>mechanism</code> parameters into one response</td>
</tr>
</tbody>
</table>

**Examples**

1. To use only the local `/etc/hosts` file for host name resolution, enter:
   ```
   hosts local
   ```

2. To use the LDAP server to resolve a host name that cannot be found in the the `/etc/hosts` file, enter:
   ```
   hosts local continue
   hosts ldap
   ```

3. To use only DNS to resolve host names and to use NIS to resolve protocols, enter:
   ```
   hosts dns
   protocols nis
   ```

4. To use only NIS to resolve host name addresses and netgroups and to use the local files to resolve services and networks, enter:
   ```
   hosts nis
   services local
   netgroup nis
   networks local
   ```

5. To try to resolve host names from the local `/etc/hosts` file and, after not finding them, try to resolve from DNS, then NIS, enter:
   ```
   hosts local continue
   hosts dns continue
   hosts nis continue
   ```

6. To try to resolve host names from the local `/etc/hosts` file, merge any information found with any DNS information found, and then merge this information to any NIS information found, enter:
   ```
   hosts local merge
   hosts dns merge
   hosts nis
   ```

   If the resolver finds no information, it returns none. If it finds information from more than one source, it returns that information as a merged response from all of the available sources.
7. To examine the local `/etc/services` file for port numbers before querying NIS, enter:

```
services local continue
services nis
```
This entry in the `/etc/irs.conf` file could speed up the request; normally, querying NIS takes more time than querying the local file. If the resolver does not find the information in the local file, it searches NIS.

8. To query for IPv4 network addresses only from DNS and to query IPv6 host addresses only from the local file, enter:

```
networks dns4
hosts local6
```

9. In this example, assume the following presuppositions:

- The `/etc/hosts` file contains the following information:
  
  1.2.3.4 host4
  1.2.3.5 host5
  1.2.3.6 host6

- The information in DNS is the following:
  
  1.2.3.2 host2
  1.2.3.3 host3

- The information in NIS is the following:
  
  1.2.3.1 host1

To instruct the `gethostbyname` subroutine to look for the host name first in the local configuration files, then to continue to search in DNS if the host name is not found, and finally to continue searching in NIS if the host name is not found, create the following entry in the `/etc/irs.conf` file:

```
hosts local continue
hosts dns continue
hosts nis
```

In this example, the `gethostbyname` subroutine cannot find the host name in the `/etc/hosts` file and continues to search for the host name in DNS. After not finding it in DNS, it continues to search in NIS. After finding the address in NIS, it returns 1.2.3.1.

10. In this example, assume the following presuppositions:

- The `/etc/hosts` file contains the following information:
  
  1.1.1.1 hostname

- The information in DNS is the following:
  
  1.1.1.2 hostname

To instruct the `gethostbyname` subroutine to merge all the answers from the specified mechanisms into one reply, create the following entry in the `/etc/irs.conf` file:

```
hosts local merge
hosts dns
```

The `gethostbyname` subroutine returns 1.1.1.1 1.1.1.2.

**Files**

- `/etc/hosts` Contains the Internet Protocol (IP) name and addresses of hosts on the local network
- `/etc/protocols` Contains official names and protocol numbers of protocol aliases
- `/etc/services` Contains lists of the port numbers, transport protocols, and names of well-known services
- `/etc/netgroup` Contains a list of groups of hosts, networks, and users in these groups
- `/etc/networks` Contains a list of network names and addresses
- `/etc/resolv.conf` Contains Domain Name Protocol (DOMAIN) name-server information for local resolver subroutines
- `/etc/netsvc.conf` Specifies the ordering of certain name resolution services
- `/etc/resolv.ldap` Contains the IP address of the LDAP server
Related Information

The hosts file format for TCP/IP, netgroup file for NIS, netsvc.conf file, networks file format for TCP/IP, protocols file format for TCP/IP, resolv.conf file format for TCP/IP, resolv.ldap file format for TCP/IP, services file format for TCP/IP.

The ldap.cfg file.

The mksecldap command.

The ypbind daemon.

The gethostbyname, gethostbyaddr and gethostent subroutines for host names.

The getnetbyname, getnetbyaddr, getnetent subroutines for networks.

The getservbyname, getservbyport, getservent subroutines for services.

The getprotobyname, getprotobynumber, getprotoent subroutines for protocols.

The getnetgrent subroutine for netgroups.

ispaths File

Purpose

Defines the location of all databases in a library.

Description

The ispaths file contains a block of information (a stanza) for each database in a library. A library consists of up to 63 standalone or cross-linked databases. The ispaths file for the default database library resides in the /usr/lpp/info/data directory. The ispaths files for other public libraries may reside in the /usr/lpp/info/data/LibraryName directory, and contain a stanza of information for each database in the library.

Each stanza must have the following format:

<table>
<thead>
<tr>
<th>Line</th>
<th>Explanation of Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Represents the number of the database. This number can be between 0 and 1462, with a maximum of 1563 databases in a library. (Database number 1563 is reserved for the help database.)</td>
</tr>
<tr>
<td>primnav TRUE</td>
<td>(Optional.) Indicates whether the database contains any of the primary navigation articles. The primnav line can be set to TRUE for only one database in the library. Omit this line unless its value is TRUE.</td>
</tr>
<tr>
<td>browse TRUE</td>
<td>(Optional.) Indicates whether the entire library is browse enabled with the browse button displayed in the reading window. Omit this line unless its value is not TRUE.</td>
</tr>
<tr>
<td>glossary TRUE</td>
<td>(Optional.) Indicates whether the database contains glossary entries. The glossary line can be set to TRUE for only one database in the library. Omit this line unless its value is TRUE.</td>
</tr>
</tbody>
</table>
name Database

title DatabaseTitle

database name

key DatabasePath/DatabaseName.key

rom DatabasePath/DatabaseName.rom

The optional field browse can be specified in any of the stanzas, and its value will be applied to the entire library. The browse field does not need to be specified in each stanza for each library that has browse capability.

Examples

The following is an example of an ispaths file for a sample database.

The isprime file for this database specifies these primary navigation articles:

- Commands
- System Calls
- Subroutines
- Special Files
- File Formats
- List of Tasks
- List of Books
- Education

All the top-level lists reside in the navigation database.

```bash
####################
# info Navigation Database
####################
id 0
primenav TRUE
browse TRUE
title Navigation
key /usr/lpp/info/%L/nav/nav.key
rom /usr/lpp/info/%L/nav/nav.rom

####################
# info System Calls Database
####################
id 1
name calls
title System Calls
key /usr/lpp/info/%L/calls/calls.key
rom /usr/lpp/info/%L/calls/calls.rom

####################
# info Subroutines Database
####################
id 2
name subs
title Subroutines
```

110 AIX 5L Version 5.3 Files Reference
<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Title</th>
<th>Key</th>
<th>Rom</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3</td>
<td>Special Files</td>
<td>/usr/lpp/info/%L/file/file.key</td>
<td>/usr/lpp/info/%L/file/file.rom</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>File Formats</td>
<td>/usr/lpp/info/%L/fls/fls.key</td>
<td>/usr/lpp/info/%L/fls/fls.rom</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>Commands</td>
<td>/usr/lpp/info/%L/cmds/cmds.key</td>
<td>/usr/lpp/info/%L/cmds/cmds.rom</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>Content Lists</td>
<td>/usr/lpp/info/%L/books/books.key</td>
<td>/usr/lpp/info/%L/books/books.rom</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>Education</td>
<td>/usr/lpp/info/%L/educ/educ.key</td>
<td>/usr/lpp/info/%L/educ/educ.rom</td>
</tr>
</tbody>
</table>

**Files**

- `/usr/lpp/info/data/ispaths`
- `/usr/lpp/info/data/LibName/ispaths`
- `/usr/lpp/info/data/LibName/isprime`

Contains the `ispaths` file for the operating system library.

Contains the `ispaths` file for the `LibName` library.

Contains the names and numbers of button labels for the primary navigation articles in `LibName`.
Related Information

The **isprime** file.

isprime File

**Purpose**
Specifies the labels for links to primary navigation articles.

**Description**
The **isprime** file specifies labels for buttons located at the bottom of a navigation window. These button labels or menu options serve as links to the primary navigation articles. Labels for up to eight primary navigation articles can be defined in the **isprime** file. The text string that serves as the label or options can consist of any alphanumeric combination, including spaces.

The format for the **isprime** file is as follows:

1. TextForFirstLink
2. TextForSecondLink
3. TextForThirdLink
4. TextForFourthLink
5. TextForFifthLink
6. TextForSixthLink
7. TextForSeventhLink
8. TextForEighthLink

**Examples**
An **isprime** file for a sample database might look as follows:

1. Commands
2. System Calls
3. Subroutines
4. Special Files
5. File Formats
6. List of Tasks
7. List of Books
8. Education

**Files**

/\usr/lpp/info/data/LibraryName/isprime

Contains labels for links to primary navigation articles.

Related Information

The **ispaths** file.

.kshrc File

**Purpose**
Contains a shell script that customizes the Korn shell environment.

**Description**
The **$HOME/.kshrc** file is a shell script that customizes the Korn-shell environment. This **.kshrc** script often contains a list of environment variables, command aliases, and function definitions that customize the Korn-shell environment.
Each time you start a new instance of the Korn shell, the `ksh` command examines the value of the `ENV` environment variable set in the `$HOME/.profile` file. If the `ENV` environment variable contains the name of an existing, readable file, the `ksh` command runs this file as a script. By convention, this file is named `$HOME/.kshrc`. You can use another name, but you must set the `ENV` environment variable to point to it.

**Note:** `.kshrc` should never output (echo, print, or call any program that echos or prints) anything.

**Examples**

The following is a sample of a `.kshrc` script on one specific system. The contents of your `.kshrc` file can be significantly different.

```bash
# @(#)kshrc 1.0
# Base Korn Shell environment
# Approach:
#   shell        initializations go in "./.kshrc
#   user         initializations go in "$HOME/.profile
#   host / all_user initializations go in /etc/profile
#   hard / software initializations go in /etc/environment
# DEBUG=y       # uncomment to report
[ "$DEBUG" ] & & echo "Entering .kshrc"
set -o allexport
# options for all shells --------------------------
# LIBPATH must be here because ksh is setuid, and LIBPATH is
# cleared when setuid programs are started, due to security hole.
LIBPATH=.:/local/lib:/lib:/usr/lib
# options for interactive shells follow-----------------
TTY=$(tty|cut -f3-4 -d/)
HISTFILE=$HOME/.sh_hist$(echo ${TTY} | tr -d '/')
PWD=$(pwd)
PS1='
${LOGNAME}@${HOSTNAME}
on ${TTY}
[${PWD}]
'
# aliases
[ "$DEBUG" ] & & echo "Setting aliases"
alias man="/afs/austin/local/bin/man -e less"
alias pg="pg -n -p':Page %d: '"
alias more="pg -n -p':Page %d: '"
alias cls="tput clear"
alias sane="stty sane"
alias rsz='eval $(resize)'
# mail check
if [ -s "$MAIL" ]       # This is at Shell startup. In
then echo "$MAILMSG"   # normal operation, the Shell checks
fi                    # periodically.
# aixterm window title
[[ "$TERM" = "aixterm" ]] & & echo
"\033[0;$USER@$(HOSTNAME	1)\007"
# functions
[ "$DEBUG" ] & & echo "Setting functions"
function pid { ps -e | grep $0 | cut -d" " -f1; }
function df {
  /bin/df $* | grep -v afs;
echo "\nAFS:";
/usr/afs/bin/fs listquota /afs;
}
```
function term {
  if [ $# -eq 1 ]
    then
      echo $TERM
      TERM=$1
      export TERM
    fi
  echo $TERM
}

function back {
  cd $OLDPWD
  echo $CWD $OLDPWD
}

[ "$DEBUG" ] && echo "Exiting .kshrc"
set +o alllexport

Files

/etc/environment Contains system-wide environment variable definitions.
/etc/profile Contains system-wide environment customization.
$HOME/.kshrc Sets the user environment for each start of the Korn shell.
$HOME/.profile Contains user-specific logon initialization.

Related Information

The ksh command.
The Shells and the Files in Operating system and device management.

lapi_subroutines Information

Purpose
Provides overview information about the subroutines that constitute the low-level application programming interface (LAPI).

Library
Availability Library (liblapi_r.a)

C Syntax

#include <lapi.h>

int lapi_subroutines(parm1, parm2...)
type1 parm1;
type2 parm2;
;

FORTRAN Syntax

#include 'lapif.h'

LAPI_SUBRoutines(parm1, parm2..., ierror)
TYPE1 :: parm1;
TYPE2 :: parm2;
;
INTEGER ierror
Description
LAPI subroutines provide a wide variety of functions that can be used efficiently and flexibly to obtain most behaviors required from any parallel programming API.

Programming with C++
LAPI subroutines provide extern “C” declarations for C++ programming.

Profiling
LAPI’s profiling interface includes wrappers for each LAPI function, so you can collect data about each of the LAPI calls. See the RSCT for AIX 5L: LAPI Programming Guide for more information.

Querying runtime values
You can find out the size (or size range) of certain parameters by calling the LAPI_Qenv subroutine with the appropriate query type. For example, call LAPI_Qenv with the LOC_ADDRTBL_SZ query type to find out the size of the address table used by the LAPI_Addr_set subroutine:

LAPI_Qenv(hndl, LOC_ADDRTBL_SZ, ret_val)

Now, if you want to register a function address using LAPI_Addr_set:

LAPI_Addr_set (hdl, addr, addr_hdl)

The value of index addr_hdl must be in the range:

1 <= addr_hdl < LOC_ADDRTBL_SZ

When used to show the size of a parameter, a comparison of values, or a range of values, valid values for the query parameter of the LAPI_Qenv subroutine appear in SMALL, BOLD capital letters. For example:

NUM_TASKS

is a shorthand notation for:

LAPI_Qenv(hndl, NUM_TASKS, ret_val)

See LAPI_Qenv subroutine for a list of the query parameter’s valid values.

Parameters
Parameter definitions are listed as follows:

INPUT
parm1
Describes parm1.

INPUT/OUTPUT
This section includes all LAPI counters.
parm2
Describes parm2.

OUTPUT
Function calls are nonblocking, so counter behavior is asynchronous with respect to the function call.
Specifies a FORTRAN return code. This is always the last parameter.

**Return Values**

**LAPI_SUCCESS**
Indicates that the function call completed successfully.

Any other return values for the subroutine appear here.

For a complete list, see the *RSCT for AIX 5L: LAPI Programming Guide*.

For information about LAPI error messages, see *RSCT: Messages*.

**Restrictions**
Any specific restrictions for the subroutine appear here.

Also, see the *RSCT for AIX 5L: LAPI Programming Guide* for more information.

**C Examples**
Any C examples of the subroutine appear here.

**FORTRAN Examples**
Any FORTRAN examples of the subroutine appear here.

**Related Information**
Any information that is related to the subroutine (including names of related subroutines) appears here.

---

**ldapid.ldif.template File**

**Purpose**
Sets the base ID entry in LDAP for new accounts.

**Description**
The `/etc/security/ldap/ldapid.ldif.template` file can be used to update the base ID entries of an LDAP server. With proper value settings to the attributes of the base ID entry, new LDAP accounts created using the `mkuser` and `mkgroup` commands will have numeric ID values greater or equal to the corresponding base value.

For example, if `aixuserid` value is set to 10000, new user accounts created in LDAP will have numeric ID values greater than or equal to 10 000.

Because specifying IDs from the command line using the `mkuser` and `mkgroup` commands is not under control of the base ID entry, an administrator can create accounts of any ID value by specifying the ID from the command line.

The base ID entry contains the following four fields:

- `aixadmingroupid`: Base ID for admin groups. The default value is 1.
- `aixadminuserid`: Base ID for admin users. The default value is 1.
- `aixgroupid`: Base ID for groups. The default value is 200.
- `aixuserid`: Base ID for users. The default value is 200.
These values can be changed by using the `ldapadd` command and `ldapmodify` command with the
`/etc/security/ldap/ldapid.ldif.template` file. The content of the file:

**Example**

dn: cn=aixbaseid,<ou=system,cn=aixdata>
objectClass: aixadmin
aixadmingroupid: 10000
aixadminuserid: 10000
aixgroupid: 10000
aixuserid: 10000

**Location**

`/etc/security/ldap/ldapid.ldif.template`

Contains the template base ID entry for LDAP servers.

**Related Information**

The "login.cfg File" on page 120.


**limits File**

**Purpose**

Defines process resource limits for users.

**Description**

**Note:** Changing the limit does not affect those processes that were started by `init`. Alternatively, `ulimits` are only used by those processes that go through the login processes.

The `/etc/security/limits` file defines process resource limits for users. This file is an ASCII file that contains stanzas that specify the process resource limits for each user. These limits are set by individual attributes within a stanza.

Each stanza is identified by a user name followed by a colon, and contains attributes in the `Attribute=Value` form. Each attribute is ended by a new-line character, and each stanza is ended by an additional new-line character. If you do not define an attribute for a user, the system applies default values.

If the hard values are not explicitly defined in the `/etc/security/limits` file but the soft values are, the system substitutes the following values for the hard limits:

<table>
<thead>
<tr>
<th>Resource</th>
<th>Hard Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core Size</td>
<td>unlimited</td>
</tr>
<tr>
<td>CPU Time</td>
<td>cpu</td>
</tr>
<tr>
<td>Data Size</td>
<td>unlimited</td>
</tr>
<tr>
<td>File Size</td>
<td>fsize</td>
</tr>
<tr>
<td>Memory Size</td>
<td>unlimited</td>
</tr>
<tr>
<td>Stack Size</td>
<td>4194304</td>
</tr>
<tr>
<td>File Descriptors</td>
<td>unlimited</td>
</tr>
</tbody>
</table>

**Note:** Use a value of -1 to set a resource to `unlimited`. 
If the hard values are explicitly defined but the soft values are not, the system sets the soft values equal to the hard values.

You can set the following limits on a user:

<table>
<thead>
<tr>
<th>Limit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fsize</td>
<td>Identifies the soft limit for the largest file a user’s process can create or extend.</td>
</tr>
<tr>
<td>core</td>
<td>Specifies the soft limit for the largest core file a user’s process can create.</td>
</tr>
<tr>
<td>cpu</td>
<td>Sets the soft limit for the largest amount of system unit time (in seconds) that a user’s process can use.</td>
</tr>
<tr>
<td>data</td>
<td>Identifies the soft limit for the largest process data segment for a user’s process.</td>
</tr>
<tr>
<td>stack</td>
<td>Specifies the soft limit for the largest process stack segment for a user’s process.</td>
</tr>
<tr>
<td>rss</td>
<td>Sets the soft limit for the largest amount of physical memory a user’s process can allocate. This limit is not enforced by the system.</td>
</tr>
<tr>
<td>nofiles</td>
<td>Sets the soft limit for the number of file descriptors a user process may have open at one time.</td>
</tr>
<tr>
<td>core_hard</td>
<td>Specifies the largest core file a user’s process can create.</td>
</tr>
<tr>
<td>cpu_hard</td>
<td>Sets the largest amount of system unit time (in seconds) that a user’s process can use.</td>
</tr>
<tr>
<td>data_hard</td>
<td>Identifies the largest process data segment for a user’s process.</td>
</tr>
<tr>
<td>fsize_hard</td>
<td>Identifies the largest file a user’s process can create or extend.</td>
</tr>
<tr>
<td>rss_hard</td>
<td>Sets the largest amount of physical memory a user’s process can allocate. This limit is not enforced by the system.</td>
</tr>
<tr>
<td>stack_hard</td>
<td>Specifies the largest process stack segment for a user’s process.</td>
</tr>
<tr>
<td>nofiles_hard</td>
<td>Sets the soft limit for the number of file descriptors a user process may have open at one time.</td>
</tr>
</tbody>
</table>

Except for the cpu and nofiles attributes, each attribute must be a decimal integer string that represents the number of 512-byte blocks allotted to a user. This decimal integer represents a 32-bit value and can have a maximum value of 2147483647. The cpu and nofiles attributes represent the maximum number of seconds of system time that a user’s process can use, and the maximum number of files a user’s process can have open at one time. For an example of a limits stanza, see the “Examples” section.

When you create a user with the mkuser command, the system adds a stanza for the user to the limits file. Once the stanza exists, you can use the chuser command to change the user’s limits. To display the current limits for a user, use the lsuser command. To remove users and their stanzas, use the rmuser command.

Note: Access to the user database files should be through the system commands and subroutines defined for this purpose. Access through other commands or subroutines may not be supported in future releases.

Security

Access Control: This file should grant read (r) access to the root user and members of the security group, and write (w) access only to the root user. Access for other users and groups depends upon the security policy for the system.

Auditing Events:

<table>
<thead>
<tr>
<th>Event</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>S_LIMITS_WRITE</td>
<td>file name</td>
</tr>
</tbody>
</table>

Examples

A typical record looks like the following example for user dhs:
dhs:
  fsize = 8192
  core = 4096
  cpu = 3600
  data = 1272
  stack = 1024
  rss = 1024
  nofiles = 2000

Files

/etc/security/limits
   Specifies the path to the file.
   Contains the basic group attributes.

/etc/group
   Contains the extended attributes of groups.

/etc/secure/group
   Contains the basic user attributes.

/etc/passwd
   Contains password information.

/etc/security/passwd
   Contains the extended attributes of users.

/etc/security/user
   Contains the environment attributes of users.

/etc/security/environ
   Contains audit-system configuration information.

/etc/security/audit/config
   Contains the default values for user accounts.

/files
   Contains last login information.

/related
   Information
   The chuser command, lsuser command, mkuser command, rmuser command.

The enduserdb subroutine, getuserattr subroutine, lDtouser subroutine, nextuser subroutine, putuserattr subroutine, setuserdb subroutine.

File and system security in Operating system and device management.

local_domain File

Purpose
Contains the NFS local domain.

Description
The /etc/nfs/local_domain file contains the local NFS domain of the system. This local domain is used to determine how to translate incoming and outgoing NFS requests. The NFS local domain must be set before using NFS V4.

The NFS local domain may be set using the chnfsdom command.

The format of the /etc/nfs/local_domain is the domain name on the first line.

Files

/etc/nfs/local_domain
   The local_domain file.

Related Information
The chnfsdom command.
### login.cfg File

#### Purpose
Contains configuration information for login and user authentication.

#### Description
The `/etc/security/login.cfg` file is an ASCII file that contains stanzas of configuration information for login and user authentication. Each stanza has a name, followed by a colon (:), that defines its purpose. Attributes are in the form `Attribute=Value`. Each attribute ends with a newline character, and each stanza ends with an additional newline character. For an example of a stanza, see the "Examples" section.

There are two types of stanzas:

<table>
<thead>
<tr>
<th>Stanzas</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>port</code></td>
<td>Defines the login characteristics of ports.</td>
</tr>
<tr>
<td><code>user configuration</code></td>
<td>Defines programs that change user attributes.</td>
</tr>
</tbody>
</table>

#### Port Stanzas
Port stanzas define the login characteristics of ports and are named with the full path name of the port. Each port should have its own separate stanza. Each stanza has the following attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>herald</code></td>
<td>Defines the login message printed when the <code>getty</code> process opens the port. The default herald is the <code>login</code> prompt. The value is a character string.</td>
</tr>
<tr>
<td><code>herald2</code></td>
<td>Defines the login message printed after a failed login attempt. The default herald is the <code>login</code> prompt. The value is a character string.</td>
</tr>
<tr>
<td><code>logindelay</code></td>
<td>Defines the delay factor (in seconds) between unsuccessful login attempts. The value is a decimal integer string. The default value is 0, indicating no delay between unsuccessful login attempts.</td>
</tr>
<tr>
<td><code>logindisable</code></td>
<td>Defines the number of unsuccessful login attempts allowed before the port is locked. The value is a decimal integer string. The default value is 0, indicating that the port cannot lock as a result of unsuccessful login attempts.</td>
</tr>
<tr>
<td><code>logininterval</code></td>
<td>Defines the time interval (in seconds) in which the specified unsuccessful login attempts must occur before the port is locked. The value is a decimal integer string. The default value is 0.</td>
</tr>
<tr>
<td><code>loginreenable</code></td>
<td>Defines the time interval (in minutes) a port is unlocked after a system lock. The value is a decimal integer string. The default value is 0, indicating that the port is not automatically unlocked.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Definition</td>
</tr>
<tr>
<td>-------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| logintimes  | Specifies the times, days, or both, the user is allowed to access the system. The value is a comma-separated list of entries of the following form:  
  
  
  \[\!:]time-time  
  -or-  
  \[\!:]day[-day][:time-time]  
  -or-  
  \[\!:]date[-date][:time-time]  

  The *day* variable must be one digit between 0 and 6 that represents one of the days of the week. A 0 (zero) indicates Sunday and a 6 indicates Saturday.  

  The *time* variable is 24-hour military time (1700 is 5:00 p.m.). Leading zeroes are required. For example, you must enter 0800, not 800. The *time* variable must be four characters in length, and there must be a leading colon (:). An entry consisting of only a time specification applies to every day. The start hour of a time value must be less than the end hour.  

  The *date* variable is a four digit string in the form *mmdd*. *mm* represents the calendar month, with 00 indicating January and 11 indicating December. *dd* represents the day number. For example 0001 represents January 1. *dd* may be 00 to indicate the entire month, if the entry is not a range, or indicating the first or last day of the month depending on whether it appears as part of the start or end of a range. For example, 0000 indicates the entire month of January, 0500 indicates the entire month of June. 0311-0500 indicates April 11 through the last day of June.  

  Entries in this list specify times that a user is allowed or denied access to the system. Entries not preceded by an exclamation point (!) allow access and are called ALLOW entries. Entries prefixed with an exclamation point (!) deny access to the system and are called DENY entries. The ! operator applies to only one entry, not the whole restriction list. It must appear at the beginning of an entry. |
| pwdprompt   | Defines the message that is displayed at a password prompt. The message value is a character string. Format specifiers will not be interpreted. If the attribute is undefined, a default prompt from the message catalog will be used. |
| sak_enabled | Defines whether the secure attention key (SAK) is enabled for the port. The SAK key is the Ctrl-X, Ctrl-R key sequence. Possible values for the sak_enabled attribute are:  

  - **true**: SAK processing is enabled, so the key sequence establishes a trusted path for the port.  
  - **false**: SAK processing is not enabled, so a trusted path cannot be established. This is the default value.  

  The sak_enabled stanza can also be modified to close a potential security exposure that exists when tty login devices are writable by others; for example, when the tty mode is 0622. If the sak_enabled stanza is set to True, the tty mode is set to a more restrictive 0600 at login. If the sak_enabled stanza is set to False (or absent), the tty mode is set to 0622. |
| synonym     | Defines other path names for the terminal. This attribute revokes access to the port and is used only for trusted path processing. The path names should be device special files with the same major and minor number and should not include hard or symbolic links. The value is a list of comma-separated path names.  

  Synonyms are not associative. For example, if you specify `synonym=/dev/tty0` in the stanza for the `/dev/console` path name, then the `/dev/tty0` path name is a synonym for the `/dev/console` path name. However, the `/dev/console` path name is not a synonym for the `/dev/tty0` path name unless you specify `synonym=/dev/console` in the stanza for the `/dev/tty0` path name. |
Attribute | Definition
--- | ---
usernameecho | Defines whether the user name is echoed on a port. Possible values for the usernameecho attribute are:

- **true**: User name echo is enabled. The user name will be displayed. This is the default value.
- **false**: User name echo is disabled. The user name will not be echoed at the login prompt and will be masked out of security related messages that contain the user name.

**User-Configuration Stanzas**
User-configuration stanzas provide configuration information for programs that change user attributes. There is one user-configuration stanza: **usw**.

**Note:** Password restrictions have no effect if you are on a network using Network Information Services (NIS). See "Network Information Service (NIS) Overview for System Management" in Networks and communication management for a description of NIS.

The **usw** stanza defines the configuration of miscellaneous facilities. The following attributes can be included:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Definition</th>
</tr>
</thead>
</table>
| auth_type | Defines the route through which all users will be authenticated (in supported applications). The two values to which **auth_type** can be set are:

- **PAM_AUTH**: Use PAM to authenticate users via the `/etc/pam.conf` file
- **STD_AUTH**: Use an application’s standard means of user authentication. This is the default value. |

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Definition</th>
</tr>
</thead>
</table>
| dist_uniqid | Defines the system configuration for resolving ID collision for creating/modifying user/group accounts among registries. The valid values to which **dist_uniqid** can be set are:

- **never**: Do not check for ID collision against the nontarget registries. This is the default setting.
- **always**: Check for ID collision against all other registries. If a collision is detected between the target registry and any other registry, account creation/modification fails. |

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>uniqbyname</td>
<td>Check for ID collision against all other registries. Collision between registries is allowed only if the account to be created has the same name as the existing account.</td>
</tr>
</tbody>
</table>

**Note:** ID collision detection in the target registry is always enforced regardless of the **dist_uniqid** attribute.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>logintimeout</td>
<td>Defines the time (in seconds) the user is given to type the login name and the password. The value is a decimal integer string. The default is a value of 60. The login session will be timed out if there is no input for login name after the timer has expired.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>maxlogins</td>
<td>Defines the maximum number of simultaneous logins to the system. The format is a decimal integer string. The default value varies depending on the specific machine license. A value of 0 indicates no limit on simultaneous login attempts. <strong>Note:</strong> Login sessions include <strong>rlogins</strong> and <strong>telnets</strong>. These are counted against the maximum allowable number of simultaneous logins by the <strong>maxlogins</strong> attribute.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>shells</td>
<td>Defines the valid shells on the system. This attribute is used by the <strong>chsh</strong> command to determine which shells a user can select. The value is a list of comma-separated full path names. The default is <code>/usr/bin/sh</code>, <code>/usr/bin/bsh</code>, <code>/usr/bin/csh</code>, <code>/usr/bin/ksh</code>, or <code>/usr/bin/tsh</code>.</td>
</tr>
</tbody>
</table>
Security

Access Control
This command should grant read (r) and write (w) access to the root user and members of the security group.

Auditing Events

<table>
<thead>
<tr>
<th>Event</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>S_LOGIN_WRITE</td>
<td>File name</td>
</tr>
</tbody>
</table>

Examples

A typical port stanza looks like the following:

```
/dev/tty0:
sak_enabled = true
herald = "login to tty0:"
```

Files

```
/etc/security/login.cfg
/etc/group
/etc/security/group
/etc/passwd
/etc/security/passwd
/etc/security/user
/etc/security/environ
/etc/security/limits
/etc/security/audit/config
/etc/security/lastlog
```

Specifies the path to the file.

Contains the basic attributes of groups.

Contains the extended attributes of groups.

Contains the basic attributes of users.

Contains password information.

Contains the extended attributes of users.

Contains the environment attributes of users.

Contains the process resource limits of users.

Contains audit system configuration information.

Contains last login information.

Related Information

The chfn command, chsec command, chsh command, login command, passwd command, pwdadm command, and su command.

The newpass subroutine.

Security in Operating system and device management.

IpacI Information

Purpose

Provides general information about protecting the least-privilege (LP) commands resource class and its resources using access controls that are provided by the resource monitoring and control (RMC) subsystem.

Description

RMC controls access to all of its resources and resource classes through access control lists (ACLs), using two different ACL implementations. The implementation that RMC uses depends on which class is involved. The two major differences between the implementations are in: 1) the mechanisms with which ACLs are viewed and modified and 2) whether ACLs are associated with individual resources.

RMC implements access controls for its resources and resource classes in the following ways:
1. Through ACLs that are defined by resource class stanzas in the ctmc.acls file.
   You can view and modify these ACLs by editing the ctmc.acls file. Use a stanza to define an ACL that applies to a class or to define an ACL that applies to all of the resources in a class.
   RMC uses this method for all of its resources and resource classes, except for the IBM.LPCommands resource class and its resources.
   For more information about the ctmc.acls file and the ACLs it defines, see the RSCT: Administration Guide.

2. Through ACLs that are associated with resources and a resource class within the RMC subsystem.
   You can view and modify these ACLs using LP commands. You can define an ACL that applies to a class or an ACL that applies to an individual resource of a class.
   RMC uses this method for the IBM.LPCommands resource class and its resources.
   This file provides information about ACLs that are specific to the IBM.LPCommands resource class and its resources.

The LP resource manager uses the IBM.LPCommands resource class to define LP resources. These resources represent commands or scripts that require root authority to run, but typically the users who need to run these commands do not have root authority. By using the LP resource manager commands, users can run commands that require root authority. The LP resource manager commands are:

- chlpcmd: Changes the read/write attribute values of an LP resource
- lphistory: Lists or clears a certain number of LP commands that were previously issued during the current RMC session
- lslpcmd: Lists information about the LP resources on one or more nodes in a domain
- mkllpcmd: Defines a new LP resource to RMC and specifies user permissions
- rmllpcmd: Removes one or more LP resources from the RMC subsystem
- runllpcmd: Runs an LP resource

For descriptions of these commands, see RSCT for AIX 5L: Technical Reference. For information about how to use these commands, see the RSCT: Administration Guide.

Because each LP resource can define a unique command, RMC implements ACLs for the IBM.LPCommands class that allow access to be controlled at the individual resource level and at the class level. RSCT provides a set of commands that you can use to list and modify the ACLs for the IBM.LPCommands class and its resources. The LP ACL commands are:

- chlpcraci: Changes the Class ACL
- chlpracl: Changes the Resource ACL
- chlpcpriacl: Changes the Resource Initial ACL
- chlpcrsacl: Changes the Resource Shared ACL
- lslpcraci: Lists the Class ACL
- lslpracl: Lists the Resource ACL
- lslpcpriacl: Lists the Resource Initial ACL
- lslpcrsacl: Lists the Resource Shared ACL
- mkllpcmd: Defines a new LP resource to RMC and specifies user permissions

For descriptions of these commands, see RSCT for AIX 5L: Technical Reference. For information about how to use these commands, see the RSCT: Administration Guide.
Basic ACL Structure
Typically, an ACL is composed of a list of ACL entries. Each ACL entry specifies an identity and a set of permissions granted to that identity. The complete list of ACL entries determines how the ACL controls access to the associated class or resource.

A resource-associated ACL can refer to another ACL instead of containing a list of ACL entries itself. When a resource-associated ACL refers to another ACL, the set of ACL entries in the referenced ACL controls access to the resource.

Types of ACLs
Four types of ACLs control access to the IBM.LPCommands class and its resources, as follows:

Class ACL
A Class ACL controls access to class operations on one node. You need to have been granted specific permissions to perform class operations, such as listing class attributes, creating class resources, and deleting class resources.

A Class ACL is composed of a list of ACL entries. The list of ACL entries controls access to class operations on the node. If the list is empty, no identity is permitted to perform class operations on the node.

When you try to perform a class operation on the IBM.LPCommands class on a node — creating a new resource, for example — RMC checks the Class ACL on that node to verify that you have the required permission to perform the operation. If you do not have the required permission, the operation is rejected.

One Class ACL exists on each node for the IBM.LPCommands class. Each node's Class ACL controls access to all IBM.LPCommands class operations on that node.

Resource ACL
A Resource ACL controls access to resource operations for one LP resource. You need to have been granted specific permissions to perform resource operations, such as listing resource attributes, modifying resource attributes, and running resource commands.

A Resource ACL can be composed of a list of ACL entries. In this case, the list of ACL entries controls access to resource operations for that resource. If the list is empty, no identity is permitted to perform resource operations for the resource.

A Resource ACL can refer to the Resource Shared ACL instead of containing a list of ACL entries itself. In this case, the list of ACL entries in the Resource Shared ACL controls access to resource operations for the resource. If the list is empty, no identity is permitted to perform resource operations for the resource.

When you try to perform a resource operation on an LP resource — running an LP command, for example — RMC first checks the Resource ACL for the selected resource to determine whether the Resource ACL contains a list of ACL entries or whether it refers to the Resource Shared ACL. If the Resource ACL has a list of ACL entries, RMC checks that list to verify that you have the required permission to perform the operation. If you do not have the required permission, the operation is rejected.

If the Resource ACL refers to the Resource Shared ACL, RMC checks the Resource Shared ACL to verify that you have the required permission to perform the operation. If you do not have the required permission, the operation is rejected.

One Resource ACL exists for each LP resource. When a Resource ACL refers to the Resource Shared ACL, the Resource Shared ACL that is being referenced is the one on the same node as the resource.

Resource Initial ACL
A Resource Initial ACL defines the initial contents of a Resource ACL created on a node.
Because a Resource Initial ACL is used to initialize Resource ACLs, a Resource Initial ACL can contain a list of ACL entries or a reference to the Resource Shared ACL.

When a new LP resource is created, its Resource ACL is initialized as specified by the Resource Initial ACL on the node.

One Resource Initial ACL exists on each node for the `IBM.LPCommands` class.

**Resource Shared ACL**

A Resource Shared ACL can control access to resource operations for multiple resources on one node.

A Resource Shared ACL is composed of a list of ACL entries. The list of ACL entries controls access to resource operations for all of the resources on the node that refer to the Resource Shared ACL. As with the other ACL types, the list of ACL entries can be empty.

To use this ACL, place ACL entries in it as you would in a Resource ACL. Then, modify the Resource ACLs on the same node to refer to the Resource Shared ACL. Using the Resource Shared ACL, you can use one list of ACL entries to control access to multiple resources on the same node.

One Resource Shared ACL exists on each node for the `IBM.LPCommands` class.

**ACL Entries**

An RMC ACL for LP commands specifies a list of ACL entries. Each ACL entry defines a user identity and that identity's user permissions. A *user identity* is an authenticated network identity. The *user permissions* specify the access that a user has to the class or to the resources.

**User Identities:** In an RMC ACL entry, the user identity can be in one of the following three forms:

1. **[host:]host_user_identifier**
   - Specifies a host user identifier. The optional *host:* keyword specifies that the user identifier can be matched against a network identifier that is provided by the host-based authentication (HBA) security mechanism. If the *host:* keyword is omitted and the entry does not take one of the other forms described, the entry is assumed to be a host user identifier. The host user identifier can be in one of the following three forms:
     a. **user_name@host_identifier**
        - Specifies a particular authenticated user. You can specify *host_identifier* in several different formats. These formats, which are the same as when the host user identifier format is specified as a host identifier alone, are described as follows.
     b. **host_identifier**
        - Specifies any authenticated user on the host identified. The host identifier can be:
          - a fully-qualified host name.
          - a short host name.
          - an IP address.
          - the RSCT node ID. This is the 16-digit hexadecimal number, for example: `0xaf58d41372c47686`.
          - the keyword LOCALHOST. This keyword is a convenient shorthand notation for the RSCT node ID of the node where the ACL exists. The LOCALHOST keyword is stored in the ACL.
          - the keyword NODEID. This keyword is a convenient shorthand notation for the RSCT node ID of the node where an ACL editing command is running. The NODEID keyword is not stored in the ACL; the node ID that the keyword represents is actually stored in the ACL.
     c. `"*"`
        - Specifies any authenticated user on any host. The asterisk (*) must be enclosed in double quotation marks when it is specified as command input.

2. **none:mapped_user_identifier**
Specifies a mapped name, as defined in the `ctsec_map.global` file or the `ctsec_map.local` file. For information about mapped user identifiers, see the *RSCT: Administration Guide*.

3. **UNAUTHENT**

   Specifies any unauthenticated user.

Some typical forms of a user identity are:

```
user@full_host_name
user@short_host_name
user@ip_address
user@node_ID
user@LOCALHOST
full_host_name
short_host_name
IP_address
node_ID
LOCALHOST
* 
```

When you are running LP commands, the `host_identifier` parameter is often expected to be the RSCT node ID. You can use the `NODEID` keyword to represent the node ID of the node on which the command runs. To determine the node ID otherwise, enter:

```
lsrsrc IBM.Host NodeIDs
```

To determine all of the node IDs in a management domain or a peer domain, enter:

```
lsrsrc -ta IBM.Host NodeIDs NodeNameList
```

The node ID is displayed in decimal format. Use the hexadecimal equivalent for the `host_identifier`, preceded by `0x`. If the nodes are in a peer domain, enter:

```
lsrpnode -i
```

The node ID is displayed in hexadecimal. To use this value in the commands, you need to precede this value with `0x`. If the `CT_CONTACT` environment variable is used to specify where the RMC session occurs, the `host_identifier` is expected to be a fully-qualified host name, a short host name, or an IP address.

**User Permissions:** The user permissions are expressed as a string of one or more characters, each representing a particular permission.

To compensate for the fine granularity of the permission set, RSCT provides two composite permissions. The `r` permission consists of individual permissions that allow "read" types of operations. The `w` permission consists of individual permissions that allow "write" types of operations. Most ACL entries will probably use these convenient composite permissions.

**The Permission Set:** The next two sections show two different views of the defined permission set. The first section describes the permission set using the composite permissions. The second section describes the permission set using the individual permissions.

**Using Composite Permissions**

```
r  Read permission.
```

- To view the resource attribute values for an LP resource, you need this permission for the LP resource.
- To view the `IBM.LPCommands` class attribute values, you need this permission for the `IBM.LPCommands` class.
You need this permission to list the LP ACLs. Therefore, this permission is meaningful for any LP ACL. Read permission consists of the q, l, e, and v permissions.

w Write permission.

- To change the resource attribute values for an LP resource, you need this permission for the LP resource.
- To change the class attribute values for the IBM.LPCommands class, you need this permission for the IBM.LPCommands class.
- To create or delete LP resources, you need this permission for the IBM.LPCommands class. Therefore, this permission is meaningful for any LP ACL. Write permission consists of the d, s, c, and o permissions.

a Administrator permission.

- To change the Resource ACL for an LP resource, you need this permission for the LP resource.
- To change the Class ACL, the Resource Initial ACL, or the Resource Shared ACL, you need this permission for the IBM.LPCommands class. Therefore, this permission is meaningful for any LP ACL.

x Execute permission. To run an LP command that is defined in an LP resource, you need this permission for the LP resource. Therefore, this permission is meaningful for the LP Resource ACL, the Resource Initial ACL, and the Resource Shared ACL.

0 No permission. This permission denies you access to an LP resource or to the IBM.LPCommands class. Therefore, this permission is meaningful for any LP ACL.

Using Individual Permissions

q Query permission.

- To query the resource attribute values for an LP resource, you need this permission for the LP resource.
- To query the class attribute values, you need this permission for the IBM.LPCommands class.
- You need this permission to list the LP ACLs. Therefore, this permission is meaningful for any LP ACL.

l Enumerate permission. To list the LP resources, you need this permission for the IBM.LPCommands class. Therefore, this permission is meaningful for the Class ACL.

e Event permission. To register, unregister, or query events, you need this permission for an LP resource or for the IBM.LPCommands class. Therefore, this permission is meaningful for any LP ACL.

v Validate permission. You need this permission to validate that an LP resource handle still exists. Therefore, this permission is meaningful for the Resource ACL, the Resource Initial ACL, and the Resource Shared ACL.

d Define and undefine permission. To create or delete LP resources, you need this permission for the IBM.LPCommands class. Therefore, this permission is meaningful for the Class ACL.

c Refresh permission. To refresh the IBM.LPCommands class configuration, you need this permission for the IBM.LPCommands class. Therefore, this permission is meaningful for the Class ACL.

s Set permission.

- To set resource attribute values for an LP resource, you need this permission for the LP resource.
- To set class attribute values, you need this permission for the IBM.LPCommands class.
Therefore, this permission is meaningful for any LP ACL.

- Online, offline, and reset permission. Because LP resources do not support the online, offline, and reset operations, this permission has no meaning in LP ACLs.

- Administrator permission.
  - To change the Resource ACL for an LP resource, you need this permission for the LP resource.
  - To change the Class ACL, the Resource Initial ACL, or the Resource Shared ACL, you need this permission for the IBM.LPCommands class.

Therefore, this permission is meaningful for any LP ACL.

- Execute permission. To run an LP command that is defined in an LP resource, you need this permission for the LP resource. Therefore, this permission is meaningful for the LP Resource ACL, the Resource Initial ACL, and the Resource Shared ACL.

- No permission. This permission denies you access to an LP resource or to the IBM.LPCommands class. Therefore, this permission is meaningful for any LP ACL.

Some permission characters have no meaning in certain types of ACLs. For example, the I permission has no meaning in a Resource ACL. A permission character that has no meaning in a certain type of ACL can be present in the ACL with no ill effect. For example, the I permission can be specified in an ACL entry of a Resource ACL. The presence of meaningless permissions in ACL entries is inevitable when the composite permissions are used.

In addition to the permissions that are granted explicitly by ACL entries, the root mapped identity always has query and administrator permission for ACL operations. If an ACL is set so that all access is denied, the root mapped identity can still be used to change the ACL, due to its implicit authority.

The system administrator needs to determine how ACLs should be defined for the IBM.LPCommands class and its resources. This depends on which operations users are required to perform.

**Security**

- To use the LP commands that change the Class ACL, the Resource Initial ACL, and the Resource Shared ACL, you must have query and administrator permission for the IBM.LPCommands class.
- To use the LP command that changes a Resource ACL for an LP resource, you must have query and administrator permission for the LP resource.
- To use the LP commands that list the Class ACL, the Resource Initial ACL, and the Resource Shared ACL, you must have query permission for the IBM.LPCommands class.
- To use the LP command that lists a Resource ACL for an LP resource, you must have query permission for the LP resource.

The **Security** section of each LP command description indicates which permissions are required for the command to run properly.

**Examples**

Some examples of how to modify the LP ACLs follow. In these examples, the commands are run on a management server for a group of nodes in a management domain. The management server is named ms_node and the managed nodes are called mc_node1, mc_node2, and so forth. In a management domain, it is most likely that the LP resources will be defined on the management server and the LP commands themselves will be targeted to the managed nodes. In these examples, the Resource Shared ACL is not used because separate permissions are desired for the individual LP resources. These examples assume that the LP resources have not yet been defined using the mklpcmd command.

1. You want to define the lpadmin ID to be the administrator for the LP commands. This ID will have the authority to modify the LP ACLs. You also want to give this ID read and write permission to be able to create, delete, and modify the LP resources. To set this up, use the root mapped identity to run these commands on the management server:
These commands define the \texttt{lpadmin} ID on the management server as having administrator, read, and write permission for the \texttt{IBM.LPCommands} class and for the Resource Initial ACL. The Resource Initial ACL is used to initialize a Resource ACL when an LP resource is created. Therefore, when an LP resource is created, the \texttt{lpadmin} ID will have administrator, read, and write permission to it.

2. The \texttt{lpadmin} ID can now create LP resources that define the LP commands that are needed. See the \texttt{mklpcmd} command for a description on how to create the LP resources. Access to the LP resources can be defined using the \texttt{mklpcmd} command or the \texttt{chlpracl} command. When the resource is created, the Resource Initial ACL is copied to the Resource ACL. To modify the Resource ACL using the \texttt{chlpracl} command so that \texttt{joe} will be able to use the \texttt{runlpcmd} command for the resource named \texttt{SysCmd1}, the \texttt{lpadmin} ID runs this command on the management server:

\begin{verbatim}
chlpracl SysCmd1 joe@LOCALHOST r
\end{verbatim}

This gives \texttt{joe} on the management server execute permission to the \texttt{SysCmd1} resource so he can use the \texttt{runlpcmd} command.

3. In this example, only the \texttt{lpadmin} ID has permission to create, delete, and modify LP resources. Use the \texttt{chlpclacl} command to let other users create and delete LP resources. In this case, they need to have write access to the class. To be able to list the resources in the \texttt{IBM.LPCommands} class, read permission is required. Read permission on a Resource ACL allows a user to view that LP resource. Write permission on a Resource ACL allows a user to modify that LP resource. To allow \texttt{joe} to view the LP resource named \texttt{SysCmd1}, the \texttt{lpadmin} ID runs this command on the management server:

\begin{verbatim}
chlpracl SysCmd1 joe@LOCALHOST r
\end{verbatim}

4. There are several nodes in a peer domain. There is an LP resource called \texttt{SysCmdB1} on node\texttt{B} for which \texttt{joe} needs execute permission. In addition, \texttt{joe} needs to have execute permission from nodes node\texttt{A}, node\texttt{B}, and node\texttt{D}. If you run the \texttt{chlpracl} command on node\texttt{B}, you can use \texttt{joe@LOCALHOST} for node\texttt{B}, but you need to determine the node IDs for node\texttt{A} and node\texttt{D}. To obtain the node IDs, enter:

\begin{verbatim}
lspnode -i
\end{verbatim}

The output will look like this:

\begin{verbatim}
Name OpState RSCTVersion NodeNum NodeID
nodeA Online 2.4.2.0 2 48ce221932ae0062
nodeB Online 2.4.2.0 1 7283cb8de374d123
nodeC Online 2.4.2.0 4 b3eda8374bc839de
nodeD Online 2.4.2.0 5 374bdce83e4ed38a
nodeE Online 2.4.2.0 2 ba74503cea374110
nodeF Online 2.4.2.0 1 4859dfbd44023e13
nodeG Online 2.4.2.0 4 6846374a0c7e773
\end{verbatim}

Then, to give \texttt{joe} the permissions as stated above, run on node\texttt{B}:

\begin{verbatim}
chlpracl SysCmd1 -l joe@LOCALHOST joe@0x48ce221932ae0062 \ joe@0x374bdce83e4ed38a x
\end{verbatim}

\textbf{Related Information}

Books: \textit{RSCT: Administration Guide}, for information about:
\begin{itemize}
\item the LP resource manager
\item how to use ACLs
\end{itemize}

\textbf{Commands:} \texttt{chlpclacl, chlpracl, chlpracI, chlprsacl, chlpcmd, lphistory, lslpclacl, lslpcmd, lslpracl, lslpracI, lslprsacl, lsqrnode, lsrsrC, mklpcmd, rmlpcmd, runlpcmd}

\textbf{Files:} \texttt{ctrmc.acls, ctsec_map.global, ctsec_map.local}
.maildelivery File for MH

Purpose
Specifies actions to be taken when mail is received.

Description
The $HOME/.maildelivery file contains a list of actions the slocal command performs on received mail. The slocal command reads the $HOME/.maildelivery file and performs the specified actions when you activate it.

Specify your own mail delivery instructions in the $HOME/.maildelivery file. Each line in the $HOME/.maildelivery file describes an action and the conditions under which the action should be performed. The following five parameters must be present in each line of the file. These parameters are separated by either commas or space characters:

Blank lines in the .maildelivery file are ignored. A # (pound sign) in the first column indicates a comment. The file is read from beginning to end, so several matches can be made with several actions. The .maildelivery file should be owned by the user, and the owner can be the only one with write access.

If the $HOME/.maildelivery file cannot be found or does not deliver the message, the /etc/mh/maildelivery file is used in the same manner. If the message has still not been delivered, it is put in the user’s mail drop. The default mail drop is the /usr/mail/$USER file.

The MH package contains four standard programs that can be run as receive-mail hooks: the rcvdist, rcvpack, rcvstore, and rcvty commands.

Parameters

Field
Specifies a header component to be searched for a pattern to match the Pattern parameter. Specify one of the following values for the Field parameter:

Component
Specify the header component you want to be searched; for example, From or cc.

*  Matches everything.

addr  Searches whatever field was used to deliver the message to you.

default  Matches only if the message has not been delivered yet.

Source
Specifies the out-of-band sender information.

Pattern
Specifies the character string to search for in the header component given by the Field parameter. For example, if you specified From in the Field parameter, the Pattern parameter might contain an address like sarah@mephisto.

The Pattern parameter is not case-sensitive. The character string matches any combination of uppercase and lowercase characters. Specify a dummy pattern if you use an * (asterisk) or specify default in the Field parameter.
Action

Specifies an action to take with the message if it contains the pattern specified in the Pattern parameter. Specify the following values:

file or >

Appends the message to the file specified with the "String" parameter. If the message can be written to the file, the action is considered successful. The Delivery-Date: header component is added to the message to indicate when the message was appended to the file.

pipe or |

Pipes the message as standard input to the command specified with the "String" parameter. The shell interprets the string. If the exit status from the command is 0, the action is considered successful. Prior to being given to the shell, the string is expanded with the following built-in variables:

$\{Address\}
Address used to deliver the message.

$\{Size\}  Size of the message in bytes.

$\{reply-to\}
Either the Reply-To: or From: header component of the message.

When a process is started with the pipe mechanism, the environment of the process is set as follows:

• User and group IDs are set to the recipient’s IDs.
• Working directory is the recipient's directory.
• The value of the umask variable is 0077.
• Process has no /dev/tty special file.
• Standard input is set to the message.
• Standard output and diagnostic output are set to the /dev/NULL special file. All other file descriptors are closed. The $USER, $HOME, and $SHELL environmental variables are set appropriately; no other environment variables exist.

The formula for determining the amount of time the process is given to execute is:

bytes in message x 60 + 300 seconds.

After that time, the process is terminated.

If the exit status of the program is 0, it is assumed that the action succeeded. Otherwise, the action is assumed unsuccessful.

qpipe or ^

Acts similarly to pipe, but executes the command directly after built-in variable expansion without assistance from the shell. If the exit status from the command is 0, the action is successful.

destroy

Destroys the message. This action always succeeds.

Result

Indicates how the action should be performed. You can specify one of the following values for this parameter:

A  Performs the action. If the action succeeds, the message is considered delivered.
R  Performs the action. Even if the action succeeds, the message is not considered delivered.
?  Performs the action only if the message has not been delivered. If the action succeeds, the message is considered delivered.
"String" Specifies the file to which the message can be appended if you use the file value for the Action parameter.

If you use the pipe or the qpipe value, the "String" parameter specifies the command to execute.

If you use the destroy value as the Action parameter, the "String" parameter is not used, but you must still include a dummy "String" parameter.

**Note:** To be notified that you have mail, you must specify the rcvttty command in the .mailedelivery file.

**Examples**

1. To save a message in a particular file, enter:
   
   From george file A george.mail

   This example directs the slocal command to search the From header line in messages. When the slocal command finds a message from george, it files the message in a file called george.mail.

2. To save a copy of a message in a file, enter:
   
   addr manager > R proj_X/statlog

   This example directs the slocal command to search the address fields in messages. When it finds a message for the project manager, the slocal command files a copy of the message in a file called proj_X/statlog. The original message is not considered delivered (the R value), so the message is still treated as mail and you will be notified as usual.

3. To be notified that you have received mail, enter:
   
   * - | R "/usr/lib/mh/rcvttty /home/sarah/allmail"

   In this example, the /home/sarah/allmail file contains the line:

   echo "You have mail"

   The /home/sarah/allmail file must have execute permission. When you have mail, the words You have mail are displayed on your console.

4. To forward a copy of a message, enter:
   
   addr manager | A "/usr/lib/mh/rcvdist amy"

   This example directs the slocal command to search the address fields in messages. When it finds a message to the project manager, the slocal command sends a copy of the message to amy. The original message is not affected. The action is always performed (the A value). The command that the slocal command reads to distribute the copy to another user is the rcvdist command.

5. To save any undelivered messages, enter:
   
   default - > ? mailbox

   This example directs the slocal command to find all undelivered messages. The - (dash) is a placeholder for the Pattern parameter. The > (greater than sign) instructs the slocal command to file the messages it finds. The ? (question mark) instructs the slocal command to respond only to undelivered messages. The name of the file to store undelivered messages is mailbox.

**Files**

- **$HOME/.forward**
  
  Searched by the sendmail command when mail is received, contains either the path of a machine to which to forward mail or a line to start the slocal command.

- **/usr/mail/SUSER**
  
  Provides the default mail drop.

- **/usr/lib/mh/slocal**
  
  Contains the slocal command that reads the .mailedelivery file.
/etc/mh/maildelivery

Contains the mail delivery instructions that the `slocal` command reads if none are specified in the `$HOME/.maildelivery` file.

$HOME/.maildelivery

Specifies mail-related actions for the `slocal` command to perform.

### Related Information

The `rcvdist` command, `rcvpck` command, `rcvstore` command, `rcvty` command, `sendmail` command, `slocal` command.

The `mtstailor` file.

---

### /usr/lib/security/methods.cfg File

**Purpose**

Contains the information for loadable authentication module configuration.

**Description**

The `/usr/lib/security/methods.cfg` file is an ASCII file that contains stanzas with loadable authentication module information. Each stanza is identified by a module name followed by a colon (:) and contains attributes in the form `Attribute=Value`. Each attribute ends with a new-line character and each stanza ends with an additional new-line character.

**Note:** If you are using Common Desktop Environment (CDE), you must restart the desktop login manager (`dtlogin`) for any changes to take effect. Restarting `dtlogin` will prevent CDE login failure by using the updated security mechanisms. Please read the `/usr/dt/README` file for more information.

Each stanza can have the following attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>domain</td>
<td>Specifies a free-format ASCII text string that is used by the loadable authentication module to select a data repository. This attribute is optional.</td>
</tr>
<tr>
<td>netgroup</td>
<td>Indicates netgroup enablement for this module. The following behaviors will be turned on:</td>
</tr>
<tr>
<td></td>
<td>1. Users defined in the <code>/etc/security/user</code> file as members of the module’s registry (for example, having <code>registry=LDAP</code> and <code>SYSTEM=LDAP</code>) will not be able to authenticate as module users. These users will now become <code>nis_module</code> users and will require native NIS netgroup membership. To fully enable <code>nis_module</code> netgroup users, corresponding entries in <code>/etc/security/user</code> must have <code>registry</code> and <code>SYSTEM</code> value removed or set to <code>compat</code>.</td>
</tr>
<tr>
<td></td>
<td>2. The registry value of <code>compat</code> is now supported. However, only <code>nis_module</code> users will show <code>compat</code> as their registry. Other users will show their absolute registry value.</td>
</tr>
<tr>
<td></td>
<td>3. The meaning of <code>registry=compat</code> will be expanded to include modules supporting netgroup. For example, if the LDAP module is netgroup enabled, <code>compat</code> will include the following registries: files, NIS and LDAP.</td>
</tr>
</tbody>
</table>
Attribute | Description
--- | ---
options | Specifies an ASCII text string containing optional values that are passed to the loadable authentication module upon initialization. The supported values for each module are described by the product documentation for that loadable authentication module.

The `options` attribute takes the following pre-defined values:

- **auth=module**
  Specifies the module to be used to perform authentication functions for the current loadable authentication module

- **authonly**
  Indicates that the loadable authentication module only performs authentication operations. User and group information must be provided by a different module, specified by the `db=` option. If not by a module, then user and group information must be provided by the local files database.

- **db=module**
  Specifies the module to be used for providing user and group information for the current loadable authentication module

- **dbonly**
  Indicates that the loadable authentication module only provides user and group information and does not perform authentication functions. Authentication operations must be performed by a different load module, specified by the `auth=` option. If the `auth=` option is not specified, all authentication operations fail.

- **netgroup**
  Indicates netgroup enabling of this module. The following behaviors will be turned on:
  1. Users defined in `/etc/security/user` as members of the module’s registry (for example, having registry=LDAP and SYSTEM=LDAP) will not be able to authenticate as module users. These users will now become nis_module users and will require native NIS netgroup membership. To fully enable nis_module netgroup users, corresponding entries in `/etc/security/user` must have `registry` and `SYSTEM` values removed or set to `compat`.
  2. The registry value of `compat` is now supported, however, only nis_module users will show `compat` as their registry value. Other users will show their absolute registry value.
  3. The meaning of registry `compat` will be expanded to include modules supporting netgroup. For example, if LDAP module is netgroup-enabled, `compat` will include the following registries: files, NIS and LDAP.

- **noprompt**
  The initial password prompt for authentication operations is suppressed. The loadable authentication module would then control all password prompting.

- **rootrequiresopw**
  Determines whether the root user is prompted for the old password for this loadable authentication module when changing another user’s password. If you want to disable the prompt of the old password, set this option to False. The default value is True.

You can only use the `auth=module` and `db=module` value strings for complex loadable authentication modules, which may require or be used with another loadable authentication module to provide new functionality.

The `authonly` and `dbonly` values are invalid for complex modules.

- **program**
  Names the load module containing the executable code that implements the loadable authentication method.

- **program_64**
  Names the load module containing the executable code that implements the loadable authentication method for 64-bit processes.
Security

Access Control: This file should grant read (r) and write (w) access to the root user only and read (r) access to the security group and all other users.

Examples

1. To indicate that the loadable authentication module is located in the file /usr/lib/security/DCE, enter:
   `program = /usr/lib/security/DCE`
2. To indicate that the loadable authentication module only should provide authentication functions, enter:
   `options = authonly`
3. The following example contains configuration information for the LDAP simple loadable authentication module:
   ```
   LDAP:
   program = /usr/lib/security/LDAP
   program_64 = /usr/lib/security/LDAP64
   ```
   The “LDAP” stanza gives the name of the module, used by the SYSTEM and registry attributes for a user. The name does not have to be the same as the file name given for the program attribute.
4. The following example contains configuration information for the KERBEROS complex loadable authentication module:
   ```
   KERBEROS:
   program = /usr/lib/security/KERBEROS
   program_64 = /usr/lib/security/KERBEROS64
   options = authonly,db=LDAP
   ```
   The “KERBEROS” stanza gives the name of the module as used by the SYSTEM and registry attributes for a user. This name does not have to be the same as the name of the file given for the program attribute. The options attribute indicates that the user and group information functions are to be performed by the module described by the “LDAP” stanza (in example 3).

Files

`/usr/lib/security/methods.cfg`
Specifies the path to the file.

`/etc/passwd`
Contains basic user attributes.

`/etc/security/user`
Contains the extended attributes of users.

`/usr/dt/README`
Contains dtlogin information.

Related Information

The `chuser`, `login`, `lsuser`, `passwd`, `su` command.

The `getauthdb` and `setauthdb` subroutines.

`Loadable Authentication Module Programming Interface` in AIX 5L Version 5.3 Kernel Extensions and Device Support Programming Concepts

mhl.format File

Purpose

Controls the output format of the mhl command.
Description

The `/etc/mh/mhl.format` file controls the output format of the `mhl` command when the `mhl` command functions as the message listing program. The `/etc/mh/mhl.format` file is the default attributes file. The `mhl.digest`, `mhl.forward`, and `mhl.reply` files must be specified before use.

Each line of the `mhl.format` file must have one of the following forms:

<table>
<thead>
<tr>
<th>Form</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>;Comment</td>
<td>Contains the comments specified by the <code>Comment</code> field that are ignored.</td>
</tr>
<tr>
<td>:ClearText</td>
<td>Contains text for output (<code>ClearText</code>). A line that contains a <code>:</code> (colon) only produces a blank output line.</td>
</tr>
<tr>
<td>Component:[Variable,...]</td>
<td>Defines the format of the specified <code>Component</code>.</td>
</tr>
<tr>
<td>Variable[Variable,...]</td>
<td>Applies the value specified by the <code>Variable</code> field only to the preceding component if the value follows that component. Lines having other formats define the global environment.</td>
</tr>
</tbody>
</table>

The entire `mhl.format` file is parsed before output processing begins. Therefore, if the global setting of a variable is defined in multiple places, the last global definition for that variable describes the current global setting.

The following table lists the `mhl.format` file variables and parameters.

Table 3. File Variables for the mhl.format File

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Width</td>
<td>integer</td>
<td>Sets the screen width or component width.</td>
</tr>
<tr>
<td>Length</td>
<td>integer</td>
<td>Sets the screen length or component length.</td>
</tr>
<tr>
<td>OffSet</td>
<td>integer</td>
<td>Indents the <code>Component</code> parameter the specified number of columns.</td>
</tr>
<tr>
<td>OverflowText</td>
<td>string</td>
<td>Outputs the <code>String</code> parameter at the beginning of each overflow line.</td>
</tr>
<tr>
<td>OverflowOffset</td>
<td>integer</td>
<td>Indents overflow lines the specified number of columns.</td>
</tr>
<tr>
<td>CompWidth</td>
<td>integer</td>
<td>Indents component text the specified number of columns after the first line of output.</td>
</tr>
<tr>
<td>Uppercase</td>
<td>flag</td>
<td>Outputs text of the <code>Component</code> parameter in all uppercase characters.</td>
</tr>
<tr>
<td>NoUppercase</td>
<td>flag</td>
<td>Outputs text of the <code>Component</code> parameter in the case entered.</td>
</tr>
<tr>
<td>ClearScreen</td>
<td>flag/G</td>
<td>Clears the screen before each page.</td>
</tr>
<tr>
<td>NoClearScreen</td>
<td>flag/G</td>
<td>Does not clear the screen before each page.</td>
</tr>
<tr>
<td>Bell</td>
<td>flag/G</td>
<td>Produces an audible indicator at the end of each page.</td>
</tr>
<tr>
<td>NoBell</td>
<td>flag/G</td>
<td>Does not produce an audible indicator at the end of each page.</td>
</tr>
</tbody>
</table>
Table 3. File Variables for the mhl.format File (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component</td>
<td>string/L</td>
<td>Uses the String parameter as the name for the specified the Component parameter instead of the string Component.</td>
</tr>
<tr>
<td>NoComponent</td>
<td>flag</td>
<td>Does not output the string Component for the specified Component parameter.</td>
</tr>
<tr>
<td>Center</td>
<td>flag</td>
<td>Centers the Component parameter on line. This variable works for one-line components only.</td>
</tr>
<tr>
<td>NoCenter</td>
<td>flag</td>
<td>Does not center the Component parameter.</td>
</tr>
<tr>
<td>LeftAdjust</td>
<td>flag</td>
<td>Strips off the leading white space characters from each line of text.</td>
</tr>
<tr>
<td>NoLeftAdjust</td>
<td>flag</td>
<td>Does not strip off the leading white space characters from each line of text.</td>
</tr>
<tr>
<td>Compress</td>
<td>flag</td>
<td>Changes new-line characters in text to space characters.</td>
</tr>
<tr>
<td>NoCompress</td>
<td>flag</td>
<td>Does not change new-line characters in text to space characters.</td>
</tr>
<tr>
<td>FormatField</td>
<td>string</td>
<td>Uses String as the format string for the specified component.</td>
</tr>
<tr>
<td>AddrField</td>
<td>flag</td>
<td>The specified Component parameter contains addresses.</td>
</tr>
<tr>
<td>DateField</td>
<td>flag</td>
<td>The specified Component parameter contains dates.</td>
</tr>
<tr>
<td>Ignore</td>
<td>unquoted string</td>
<td>Does not output component specified by String.</td>
</tr>
</tbody>
</table>

Variables that have integer or string values as parameters must be followed by an = (equal sign) and the integer or string value (for example, overflowoffset=5). String values must also be enclosed in double quotation marks (for example, overflowtext="***"). A parameter specified with the /G suffix has global scope. A parameter specified with the /L suffix has local scope.

Examples

The following is an example of a line that could be displayed in the mhl.format file:

```
width=80,length=40,clearscreen,overflowtext="***",overflowoffset=5
```

This format line defines the screen size to be 80 columns by 40 rows, and specifies the screen should be cleared before each page (clearscreen). The overflow text should be flagged with the *** string, and the overflow indentation should be 5 columns.

Files

/etc/mh/mhl.format Specifies the path of the mhl.format file.
Related Information
The \texttt{ap} command, \texttt{dp} command, \texttt{mhl} command, \texttt{scan} command.

\section{.mh_profile File}

\subsection{Purpose}
Customizes the Message Handler (MH) package.

\subsection{Description}
Each user of the MH package is expected to have a $\texttt{HOME/.mh_profile}$ file in the home directory. This file contains a set of user parameters used by some or all of the MH programs. Each line of the file has the following format:

\texttt{Profile-Entry: Value}

\subsection{Profile Entries}
This table describes the profile entry options for the \texttt{.mh_profile} file. Only \texttt{Path:} is required. Each profile entry is stored in either the \texttt{.mh_profile} file or the $\texttt{UserMHDirectory/context}$ file.

\begin{table}[h]
\centering
\begin{tabular}{ |l|l|l| }
\hline
\textbf{Profile Entry and Description} & \textbf{Storage File} & \textbf{Default Value} \\
\hline
Path: & \texttt{mh_profile} & \texttt{None} \\
\texttt{UserMHDirectory} & & \\
\texttt{Mail} & & \\
\hline
context: & \texttt{mh_profile} & \texttt{UserMHDirectory/context} \\
\texttt{MH context file} & & \\
\hline
Current- Folder: & \texttt{context} & \texttt{inbox} \\
\texttt{Tracks the current open folder} & & \\
\hline
Previous- Sequence: & \texttt{mh_profile} & \texttt{None} \\
\texttt{Messages or Message sequences parameter given to the program} & & \\
\texttt{For each name given, the sequence is set to 0. Each message is added to the sequence. If not present or empty, no sequences are defined.} & & \\
\hline
Sequence- Negation: & \texttt{mh_profile} & \texttt{None} \\
\texttt{The string negating a sequence when prefixed to the name of that sequence. For example, if set to \texttt{not}, \texttt{not seen} refers to all the messages that are not a member of the sequence seen.} & & \\
\hline
Unseen- Sequence: & \texttt{mh_profile} & \texttt{None} \\
\texttt{The sequences defined as messages recently incorporated by the \texttt{inc} command. For each name given, the sequence is set to 0. If not present, or empty, no sequences are defined.} & & \\
\texttt{Note: The show command removes messages from this sequence after viewing.} & & \\
\hline
\end{tabular}
\end{table}
<table>
<thead>
<tr>
<th>Profile Entry and Description</th>
<th>Storage File</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>.mh_sequences</strong>: The file, in each folder, defining public sequences. To disable the use of public sequences, leave the value of this entry blank.</td>
<td>mh_profile</td>
<td>.mh_sequences</td>
</tr>
<tr>
<td><strong>atr- SequenceFolder</strong>: Tracks the specified sequence in the specified folder.</td>
<td>context</td>
<td>None</td>
</tr>
<tr>
<td><strong>Editor</strong>: The editor to be used by the comp, dist, forw, and repl commands.</td>
<td>mh_profile</td>
<td>prompter</td>
</tr>
<tr>
<td><strong>Msg-Protect</strong>: Defines octal protection bits for message files. The <code>chmod</code> command explains the default values.</td>
<td>mh_profile</td>
<td>0644</td>
</tr>
<tr>
<td><strong>Folder- Protect</strong>: Defines protection bits for folder directories. The <code>chmod</code> command explains the default values.</td>
<td>mh_profile</td>
<td>0711</td>
</tr>
<tr>
<td><strong>Program</strong>: Sets default flags to be used when the MH program specified by the MH program field is started. For example, override the Editor: profile entry when replying to messages by entering: repl: -editor /usr/bin/ed</td>
<td>mh_profile</td>
<td>None</td>
</tr>
<tr>
<td><strong>LastEditor-next</strong>: The default editor after the editor specified by the Editor: field has been used. This takes effect at the What now? field of the comp, dist, forw, and repl commands. If you enter the editor command without a parameter to the What now? field, the editor specified by the LastEditor-next: field is used.</td>
<td>mh_profile</td>
<td>None</td>
</tr>
<tr>
<td><strong>Folder-Stack</strong>: The contents of the folder stack of the folder command.</td>
<td>context</td>
<td>None</td>
</tr>
</tbody>
</table>
### Table 4. Profile Entry Options for the .mh_profile File (continued)

<table>
<thead>
<tr>
<th>Profile Entry and Description</th>
<th>Storage File</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Alternate- Mailboxes:</strong></td>
<td>mh_profile</td>
<td>$LOGNAME</td>
</tr>
</tbody>
</table>
| Indicates your address to the repl and scan commands. The repl command is given the addresses to include in the reply. The scan command is informed the message originated from you. Host names should be the official host names for the mailboxes you indicate. Local nicknames for hosts are not replaced with their official site names. If a host is not given for a particular address, that address on any host is considered to be your current address. Enter an * (asterisk) at either end or both ends of the host mailbox to indicate pattern matching.  
**Note:** Addresses must be separated by a comma. | mh_profile | None |
| Draft-Folder:                 | mh_profile | None |
| Indicates a default draft folder for the comp, dist, forw, and repl commands. | mh_profile | None |
| **digest-issue- List:**      | context     | None |
| Indicates to the forw command the last issue of the last volume sent for the digest List. | context | None |
| **digest-volume- List:**     | context | None |
| Indicates to the forw command the last volume sent for the digest List. | context | None |
| MailDrop:                    | mh_profile   | /usr/mail/$USER |
| Indicates to the inc command your mail drop, if different from the default. This is superseded by the $MAILDROP environment variable. | mh_profile | None |
| Signature:                   | mh_profile   | None |
| Indicates to the send command your mail signature. This is superseded by the $SIGNATURE environment variable. | mh_profile | None |

### Profile Elements

The following profile elements are used whenever an MH program starts another program. You can use the .mh_profile file to select alternate programs.

<table>
<thead>
<tr>
<th>Profile Element</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>fileproc:</td>
<td>/usr/bin/refile</td>
</tr>
<tr>
<td>incproc:</td>
<td>/usr/bin/inc</td>
</tr>
<tr>
<td>installproc:</td>
<td>/usr/lib/mh/install-mh</td>
</tr>
<tr>
<td>lproc:</td>
<td>/usr/bin/more</td>
</tr>
<tr>
<td>mailproc:</td>
<td>/usr/bin/mhmail</td>
</tr>
<tr>
<td>mhlproc:</td>
<td>/usr/lib/mh/mhl</td>
</tr>
<tr>
<td>moreproc:</td>
<td>/usr/bin/more</td>
</tr>
<tr>
<td>mshproc:</td>
<td>/usr/bin/msh</td>
</tr>
<tr>
<td>packproc:</td>
<td>/usr/bin/packf</td>
</tr>
</tbody>
</table>
Profile Element | Path
--- | ---
postproc: | /usr/lib/mh/spost
rmmproc: | None
rmfproc: | /usr/bin/rmf
sendproc: | /usr/bin/send
showproc: | /usr/bin/more
whatnowproc: | /usr/bin/whatnow
whomproc: | /usr/bin/whom

Environment Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$MH</td>
<td>Specifies a profile for an MH program to read. When you start an MH program, it reads the <code>.mh_profile</code> file by default. Use the $MH environment variable to specify a different profile. If the file of the $MH environment variable does not begin with a / (slash), it is presumed to start in the current directory. The / indicates the file is absolute.</td>
</tr>
<tr>
<td>$MHCONTEXT</td>
<td>Specifies a context file that is different from the normal context file specified in the MH profile. If the value of the $MHCONTEXT environment variable is not absolute, it is presumed to start from your MH directory.</td>
</tr>
<tr>
<td>$MAILDROP</td>
<td>Indicates to the <code>inc</code> command the default mail drop. This supersedes the MailDrop: profile entry.</td>
</tr>
<tr>
<td>$SIGNATURE</td>
<td>Specifies your mail signature to the <code>send</code> and <code>post</code> commands. This supersedes the Signature: profile entry.</td>
</tr>
<tr>
<td>$HOME</td>
<td>Specifies your home directory to all MH programs.</td>
</tr>
<tr>
<td>$TERM</td>
<td>Specifies your terminal type to the MH package. In particular, these environment variables tell the <code>scan</code> and <code>mh</code> commands how to clear your terminal, and give the width and length of your terminal in columns and lines, respectively.</td>
</tr>
<tr>
<td>$editalt</td>
<td>Specifies an alternate message. This is set by the <code>dist</code> and <code>repl</code> commands during edit sessions so you can read the distributed message or the answered message. This message is also available through a link called @ (at sign) in the current directory, if your current directory and the message folder are on the same file system.</td>
</tr>
<tr>
<td>$mhdraft</td>
<td>Specifies the path name of the working draft.</td>
</tr>
<tr>
<td>$mhfolder</td>
<td>Specifies the folder containing the alternate message. This is set by the <code>dist</code> and <code>repl</code> commands during edit sessions, so you can read other messages in the current folder besides the one being distributed. The $mhfolder environment variable is also set by the <code>show</code>, <code>prev</code>, and <code>next</code> commands for use by the <code>mhl</code> command.</td>
</tr>
</tbody>
</table>

Examples

The following example has the mandatory entry for the `Path:` field. The option `-alias aliases` is used when both the `send` and `ali` commands are started. The `aliases` file resides in the mail directory. The message protection is set to 600, which means that only the user has permission to read the message files. The signature is set to Dan Carpenter, and the default editor is `vi`.

Path: Mail
send: -alias aliases
ali: -alias aliases
Msg-Protect: 600
Signature: Dan Carpenter
Editor: /usr/bin/vi

Files

- `$HOME/.mh_profile` Contains the user profile.
- `UserMHDirectory/context` Contains the user context file.
- `Folder/.mh_sequences` Contains the public sequences for the folder specified by the `Folder` variable.
mibII.my File

Purpose
Provides sample input to the mosy command.

Description
The /usr/samples/snmpd/mibII.my file is a sample input file to the mosy command, which creates an objects definition file for use by the snmpinfo command. This file is part of Simple Network Management Protocol Agent Applications in Network Support Facilities. The mosy compiler requires its input file to contain the ASN.1 definitions as described in the Structure and Identification of Management Information (SMI) RFC 1155 and the Management Information Base (MIB) RFC 1213. The mibII.my file contains the ASN.1 definitions from the MIB RFC 1213 (MIB II). RFC is the abbreviation for Request for Comments.

Comments are specified by - - (two dashes). A comment can begin at any location after the comment sign and extend to the end of the line.

The mibII.my file begins with a definition of the SNMP subtree of the MIB, as assigned by the Internet Activities Board (IAB). This definition contains the name of the RFCs from which the ASN.1 definitions are obtained.

RFC1213-MIB {iso org(3) dod(6) internet(1) mgmt(2) 1 }
DEFINITIONS ::= BEGIN
IMPORTS
    mgmt, NetworkAddress, IpAddress,
    Counter, Gauge, TimeTicks
FROM RFC1155-SMI
OBJECT-TYPE
    from RFC-1213;

mib-2 OBJECT IDENTIFIER ::= { mgmt 1 }-- MIB-II

system OBJECT IDENTIFIER ::= { mib-2 1 }
interfaces OBJECT IDENTIFIER ::= { mib-2 2 }
at OBJECT IDENTIFIER ::= { mib-2 3 }
ip OBJECT IDENTIFIER ::= { mib-2 4 }
icmp OBJECT IDENTIFIER ::= { mib-2 5 }/tcp OBJECT IDENTIFIER ::= { mib-2 6 }
udp OBJECT IDENTIFIER ::= { mib-2 7 }
egp OBJECT IDENTIFIER ::= { mib-2 8 }
-- cmot OBJECT IDENTIFIER ::= { mib-2 9 }
transmission OBJECT IDENTIFIER ::= { mib-2 10 }
snmp OBJECT IDENTIFIER ::= { mib-2 11 }

The file must contain the ASN.1 definition for each MIB variable. The ASN.1 definition is presented in an OBJECT-TYPE macro.

Following is the format of an OBJECT-TYPE macro:
The following definitions describe the pieces of the macro:

**Macro**

**ObjectDescriptor**
Indicates the textual name assigned to the MIB variable being defined. See RFC 1155 for the definition of the ObjectDescriptor variable.

**ObjectSyntax**
Indicates the abstract syntax for the object type. It must be one of:
- INTEGER
- OCTET STRING or DisplayString
- OBJECT IDENTIFIER
- NULL
- Network Address
- Counter
- Gauge
- TimeTicks
- Opaque

See RFC 1155 for definitions of each ObjectSyntax variable.

**AccessMode**
Specifies the permissions of the object, which can be either:
- read-only
- read-write
- write-only
- not-accessible

See RFC 1155 for definitions of each AccessMode variable.

**StatusType**
Specifies the status of the object, which can be either:
- mandatory
- optional
- deprecated
- obsolete

See RFC 1155 for definitions of each StatusType variable.

**Description**
Specifies a textual description of the purpose of the MIB variable being defined.

**ObjectGroup**
Defines the object group for this MIB variable. The ObjectGroup variable identifies the subtree for the MIB variable. See RFC 1213 for information on object groups.

**Entry**
Defines the unique location of the MIB variable in the ObjectGroup variable.

The ObjectGroup and Entry variables are used to specify the unique numerical object identifier for each MIB variable. See RFC 1155 for an explanation of the object identifier.

See RFC 1155 for further information on the OBJECT-TYPE macro.

This sample mibII.my file was created by extracting the definitions from Chapter 6, "Definitions," of RFC 1213. This file is shipped as /usr/samples/snmpd/mibII.my.

**Examples**
The following example of an OBJECT-TYPE macro describes the sysDescr managed object:
sysDescr OBJECT-TYPE
SYNTAX DisplayString (SIZE (0..255))
ACCESS read-only
STATUS mandatory
DESCRIPTION A textual description of the entity.
This value should include the full name and
version identification of system's hardware
type, software operating-system, and networking
software. It is mandatory that this only
contain printable ASCII characters.
::= { system 1 }

Related Information
The mosy command, snmpinfo command.

The smi.my file.

Management Information Base (MIB) and Terminology Related to Management Information Base (MIB)
Variables in AIX 5L Version 5.3 Communications Programming Concepts.

RFC 1155, RFC 1213.


mkuser.default File

Purpose
Contains the default attributes for new users.

Description
The /usr/lib/security/mkuser.default file contains the default attributes for new users. This file is an ASCII
file that contains user stanzas. These stanzas have attribute default values for users created by the
mkuser command. Each attribute has the Attribute=Value form. If an attribute has a value of $USER, the
mkuser command substitutes the name of the user. The end of each attribute pair and stanza is marked
by a new-line character.

There are two stanzas, user and admin, that can contain all defined attributes except the id and admin
attributes. The mkuser command generates a unique id attribute. The admin attribute depends on
whether the -a flag is used with the mkuser command.

For a list of the possible user attributes, see the chuser command.
Security
Access Control: If read (r) access is not granted to all users, members of the security group should be
given read (r) access. This command should grant write (w) access only to the root user.

Examples
A typical user stanza looks like the following:

user:
  pgroup = staff
  groups = staff
  shell = /usr/bin/ksh
  home = /home/$USER
  auth1 = SYSTEM

Files
/usr/lib/security/mkuser.default Specifies the path to the file.

Related Information
The chuser command, mkuser command.

User Accounts in Security.

mtstailor File for MH

Purpose
Tailors the Message Handler (MH) environment to the local environment.

Description
The entries located in the /etc/mh/mtstailor file specify how MH commands work. The following list
describes the file entries and their default values. All of the file entries are optional.

<table>
<thead>
<tr>
<th>Entry</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>localname:</td>
<td>Specifies the host name of the local system. If this entry is not defined, MH queries the</td>
</tr>
<tr>
<td></td>
<td>system for the default value.</td>
</tr>
<tr>
<td>systemname:</td>
<td>Specifies the host name of the local system in the UUCP domain. If this entry is not defined,</td>
</tr>
<tr>
<td></td>
<td>MH queries the system for the default value.</td>
</tr>
<tr>
<td>mmdfldir:</td>
<td>Specifies the location of mail drops. If this entry is present and empty, mail drops are</td>
</tr>
<tr>
<td></td>
<td>located in the user's $HOME directory. If this entry does not exist, mail drops are located in</td>
</tr>
<tr>
<td></td>
<td>the /usr/mail directory.</td>
</tr>
<tr>
<td>mmdflfil1:</td>
<td>Specifies the name of the file used as the mail drop. If this entry is not defined, the default</td>
</tr>
<tr>
<td></td>
<td>file name is the same as the user name.</td>
</tr>
<tr>
<td>mmdelim1:</td>
<td>Specifies the beginning-of-message delimiter for mail drops. The default value is four Ctrl +</td>
</tr>
<tr>
<td></td>
<td>A key sequences followed by a new-line character (. 001. 001. 001. 012). A Ctrl + A key</td>
</tr>
<tr>
<td></td>
<td>sequence is a nonprintable character not displayed on the screen.</td>
</tr>
<tr>
<td>mmdelim2:</td>
<td>Specifies the end-of-message delimiter for mail drops. The default value is four Ctrl + A key</td>
</tr>
<tr>
<td></td>
<td>sequences followed by a new-line character (. 001. 001. 001. 012). A Ctrl + A key sequence is</td>
</tr>
<tr>
<td></td>
<td>a nonprintable character not displayed on the screen.</td>
</tr>
<tr>
<td>Entry</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>mmailid:</td>
<td>Specifies whether support for the <code>MMailID</code> variable in the <code>/etc/passwd</code> file is enabled. If the <code>mmailid:</code> entry is set to a nonzero value, support is enabled. The <code>pw_gecos:</code> field in the <code>/etc/passwd</code> file has the following form:</td>
</tr>
<tr>
<td></td>
<td>My Full Name MailID</td>
</tr>
<tr>
<td></td>
<td>When support for the <code>MMailID</code> variable is enabled, the internal MH routines that deal with user and full names return the <code>MailID</code> variable and the <code>My Full Name</code>, respectively. The default value is 0.</td>
</tr>
<tr>
<td>lockstyle:</td>
<td>Specifies the locking discipline. A value of 0 (zero) uses the lockf system call to perform locks. A value of 1 creates lock names by appending <code>.lock</code> to the name of the file being locked. The default is 0 (zero).</td>
</tr>
<tr>
<td>lockldir:</td>
<td>Specifies the directory for locked files. The default value is the <code>/etc/locks</code> file.</td>
</tr>
<tr>
<td>sendmail:</td>
<td>Specifies the path name of the <code>sendmail</code> command. The default value is the <code>/usr/lib/sendmail</code> file.</td>
</tr>
<tr>
<td>maildelivery:</td>
<td>Specifies the path name of the file containing the system default mail delivery instructions. The default value is the <code>/etc/mh/maildelivery</code> file.</td>
</tr>
<tr>
<td>everyone:</td>
<td>Specifies the users to receive messages addressed to everyone. All users having UIDs greater than the specified number (not inclusive) receive messages addressed to everyone. The default value is 200.</td>
</tr>
</tbody>
</table>

**Files**

- `/etc/mh/mtstailor` Contains MH command definitions.

**Related Information**

The `sendmail` command.

The `.maildelivery File for MH` file, `/etc/passwd` file.

**mrouted.conf File**

**Purpose**

Default configuration information for the multicast routing daemon `mrouted`.

**Description**

The `/etc/mrouted.conf` configuration file contains entries that provide configuration information used by `mrouted`. You can specify any combination of these entries in this file.

The file format is free-form; white space and newline characters are not significant. The `phyint`, `tunnel`, and `name` entries can be specified more than once. The `boundary` and `altnet` values can be specified as many times as necessary.

The following entries and their options can be used in the `mrouted.conf` file:

**phyint** `local_addr [disable] [metric m] [threshold f] [rate_limit b] [boundary (boundary_name | scoped_addr[mask_len])] [altnet network[mask_len]]`

The `phyint` entry can be used to disable multicast routing on the physical interface identified by the local IP address `local_addr`, or to associate a non-default metric or threshold with the specified physical interface. The local IP address can be replaced by the interface name (for example, `le0`). If a physical interface is attached to multiple IP subnets, describe each additional subnet with the `altnet` option. `Phyint` entries must precede `tunnel` entries.
The options for the **phyint** entry and the actions they generate are as follows:

**local_addr**

Specifies the local address, using either an IP address or an interface name, such as `en0`.

**disable**

Disables multicast routing on the physical interface identified by `local_addr`.

**metric m**

Specifies the "cost" associated with sending a datagram on the given interface or tunnel. This option can be used to influence the choice of routes. The default value of `m` is 1. Metrics should be kept as small as possible, because `mrouted` cannot route along paths with a sum of metrics greater than 31.

**threshold t**

Specifies the minimum IP time-to-live (TTL) required for a multicast datagram to be forwarded to the given interface or tunnel. This option controls the scope of multicast datagrams. (The TTL of forwarded packets is compared only to the threshold, it is not decremented by the threshold.) The default value of `t` is 1. In general, all `mrouted` daemons connected to a particular subnet or tunnel should use the same metric and threshold for that subnet or tunnel.

**rate_limit b**

Specifies a bandwidth in Kilobits/second, which is allocated to multicast traffic. The default value of `b` is 500 Kbps on tunnels, and 0 (unlimited) on physical interfaces.

**boundary boundary_name|scoped_addr/mask_len**

Configures an interface as an administrative boundary for the specified scoped address. Packets belonging to this address are not forwarded on a scoped interface. The `boundary` option accepts either a boundary name or a scoped address and mask length. The `boundary_name` is the name assigned to a boundary with the `name` entry. The `scoped_addr` value is a multicast address. The `mask_len` value is the length of the network mask.

**altnet network/mask_len**

Specifies an additional subnet (`network`) attached to the physical interface described in the `phyint` entry. `mask_len` is the length of the network mask.

**tunnel local_addr remote_addr [metric m] [threshold t] [rate_limit b] [boundary {boundary_name|scoped_addr/mask_len}] [altnet network/mask_len]**

The `tunnel` entry can be used to establish a tunnel link between the local IP address (`local_addr`) and the remote IP address (`remote_addr`), and to associate a non-default metric or threshold with that tunnel. The local IP address can be replaced by the interface name (for example, `le0`). The remote IP address can be replaced by a host name, if and only if the host name has a single IP address associated with it. The tunnel must be set up in the `mrouted.conf` files of both routers before it can be used. The `phyint` entry can be used to disable multicast routing on the physical address interface identified by the local IP address `local_addr`, or to associate a non-default metric or threshold with the specified physical interface. The local IP address can be replaced by the interface name (for example, `le0`). If a physical interface is attached to multiple IP subnets, describe each additional subnet with the `altnet` option. Phyint entries must precede `tunnel` entries.

For a description of the options used with the `tunnel` entry, see the preceding option descriptions in the `phyint` entry.

**cache_lifetime ct**

The `cache_lifetime` entry determines the amount of time that a cached multicast route stays in the kernel before timing out. The value of `ct` is in seconds, and should lie between 300 (five minutes) and 86400 (one day). The default value is 300 seconds.
pruning state

The pruning entry enables mrouted to act as a non-pruning router. The value of state can be either on or off. You should configure your router as a non-pruning router for test purposes only. The default mode is on, which enables pruning.

name boundary_name scoped_addr/mask-len

The name entry lets you assign names to boundaries to make it easier to configure. The boundary option on the phyint and tunnel entries accepts either a boundary name or a scoped address. The boundary_name is the name you want to give to the boundary. The scoped_addr value is a multicast address. The mask_len value is the length of the network mask.

Example

This example shows a configuration for a multicast router at a large school.

```
# mrouted.conf
# Name our boundaries to make it easier
name LOCAL 239.255.0.0/16 name EE 239.254.0.0/16
#
# le1 is our gateway to compsci, don't forward our
# local groups to them
phyint le1 boundary LOCAL
#
# le2 is our interface on the classroom network,
# it has four different length subnets on it.
# Note that you can use either an IP address or an
# interface name
phyint 172.16.12.38 boundary EE altnet 172.16.15.0/26
       altnet 172.16.15.128/26 altnet 172.16.48.0/24
#
# atm0 is our ATM interface, which doesn't properly
# support multicasting
phyint atm0 disable
#
# This is an internal tunnel to another EE subnet.
# Remove the default tunnel rate limit, since this tunnel
# is over ethornets
# Don't use eroute, because it is not supposed to
# # Do this on 192.168.5.4
# tunnel 192.168.5.4 192.168.55.101 metric 1 threshold 1
# rate_limit 0
# This is our tunnel to the outside world.
# Don't use eroute, because it is not supposed to
# # Do this on 192.168.5.4
# tunnel 192.168.5.4 10.11.12.13 metric 1 threshold 32
## boundary LOCAL boundary EE
```

netgroup File for NIS

Purpose

Lists the groups of users on the network.

Description

The /etc/netgroup file defines network-wide groups. This file is used for checking permissions when doing remote mounts, remote logins, and remote shells. For remote mounts, the information in the netgroup file is used to classify machines. For remote logins and remote shells, the file is used to classify users. Each line of the netgroup file defines a group and is formatted as follows:

```
Groupname Member1 Member2 ...
```

where Member is either another group name or consists of three entries as follows:

```
hostname, username, domainname
```
Any of these three fields can be empty, in which case it signifies a wild card. The *universal* (, , ) field defines a group to which everyone belongs.

Field names that begin with something other than a letter, digit or underscore (such as -) work in precisely the opposite fashion. For example, consider the following entries:

```
justmachines  (analytica, -, ibm)
justpeople    (-, babbage, ibm)
```

The machine *analytica* belongs to the group *justmachines* in the domain *ibm*, but no users belong to it. Similarly, the user *babbage* belongs to the group *justpeople* in the domain *ibm*, but no machines belong to it.

A gateway machine should be listed under all possible host names by which it may be recognized:

```
wan (gateway , , ) (gateway-ebb, , )
```

The *domainname* field refers to the domain *n* in which the triple is valid, not the name containing the trusted host.

**Examples**
The following is an excerpt from a *netgroup* file:

```
machines  (venus, -, star)
people    (-, bob, star)
```

In this example, the machine named *venus* belongs to the group *machines* in the *star* domain. Similarly, the user *bob* belongs to the group *people* in the *star* domain.

**Files**

```
/etc/netgroup     Specifies the path of the file.
```

**Related Information**
The `makedbm` command.

The `ypserv` daemon.

Network File System Overview and Network Information Service Overview in *Networks and communication management*.

List of NIS Programming References in *Networks and communication management*.

---

**netmasks File for NIS**

**Purpose**
Contains network masks used to implement Internet Protocol (IP) standard subnetting.

**Description**
The `/etc/netmasks` file contains network masks used to implement IP standard subnetting. This file contains a line for each network that is subnetted. Each line consists of the network number, any number of spaces or tabs, and the network mask to use on that network. Network numbers and masks may be specified in the conventional IP . (dot) notation (similar to IP host addresses, but with zeroes for the host part). The following number is a line from a *netmask* file:
This number specifies that the Class B network 128.32.0.0 has 8 bits of subnet field and 8 bits of host field, in addition to the standard 16 bits in the network field. When running network information service, this file on the master is used for the netmasks.byaddr map.

### Files

/\etc/netmasks

Specifies the path of the file.

---

### Related Information

- [Network File System Overview](#) in *Networks and communication management*.
- [Network Information Service Overview](#) in *Networks and communication management*.

---

### netsvc.conf File

#### Purpose

Specifies the ordering of certain name resolution services.

#### Description

The /\etc/netsvc.conf file is used to specify the ordering of name resolution for the sendmail command, gethostbyname subroutine, gethostaddr subroutine, and gethostent subroutine and alias resolution for the sendmail command.

Several mechanisms for resolving host names and aliases are available. The gethostbyname, gethostbyaddr, and gethostent subroutines use these mechanisms for resolving names. A default order exists in which the resolver subroutines try the mechanisms for resolving host names and Internet Protocol (IP) addresses.

### Resolving Host Names

You can override the default order and the order given in the /\etc/irs.conf file by creating the /\etc/netsvc.conf configuration file and specifying the desired ordering. To specify this host ordering, create an entry in the following format:

```
hosts = value [, value]
```

Use one or more of the following values for the hosts keyword:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>auth</td>
<td>Designates the specified server as authoritative. A resolver does not continue searching for host names further than an authoritative server. For example, when two services are given as values for the host keyword and the first service is made authoritative, and if the resolver cannot find the host name in the authoritative service, then the resolver terminates its search. However, the auth option has no effect if the resolver is unable to contact the authoritative server; in this case, the resolver continues to search the next service given in the same entry.</td>
</tr>
<tr>
<td>bind</td>
<td>Uses BIND/DNS services for resolving names</td>
</tr>
<tr>
<td>local</td>
<td>Searches the local /etc/hosts file for resolving names</td>
</tr>
<tr>
<td>nis</td>
<td>Uses NIS services for resolving names. NIS must be running if you specify this option</td>
</tr>
<tr>
<td>nis+</td>
<td>Uses NIS+ services for resolving names. NIS+ must be running if you specify this option</td>
</tr>
</tbody>
</table>

**Note:** The auth option is only valid when used in conjunction with a service value for the host keyword.
<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ldap</td>
<td>Uses LDAP services for resolving names. This option works if LDAP server schema is IBM Secureway Directory compliant. <strong>Note:</strong> Although still supported, the use of <em>ldap</em> mechanism is deprecated. Use of <em>nis_ldap</em> mechanism instead is recommended.</td>
</tr>
<tr>
<td>nis_ldap</td>
<td>Uses LDAP services for resolving names. This option works if LDAP server schema is RFC 2307 compliant.</td>
</tr>
<tr>
<td>bind4</td>
<td>Uses BIND/DNS services for resolving only IPv4 addresses</td>
</tr>
<tr>
<td>bind6</td>
<td>Uses BIND/DNS services for resolving only IPv6 addresses</td>
</tr>
<tr>
<td>local4</td>
<td>Searches the local <em>/etc/hosts</em> file for resolving only IPv4 addresses</td>
</tr>
<tr>
<td>local6</td>
<td>Searches the local <em>/etc/hosts</em> file for resolving only IPv6 addresses</td>
</tr>
<tr>
<td>nis4</td>
<td>Uses NIS services for resolving only IPv4 addresses</td>
</tr>
<tr>
<td>nis6</td>
<td>Uses NIS services for resolving only IPv6 addresses</td>
</tr>
<tr>
<td>ldap4</td>
<td>Uses LDAP services for resolving only IPv4 addresses</td>
</tr>
<tr>
<td>ldap6</td>
<td>Uses LDAP services for resolving only IPv6 addresses</td>
</tr>
</tbody>
</table>

The environment variable `NSORDER` overrides the host settings in the */etc/netsvc.conf* file, which in turn overrides the host settings in the */etc/irs.conf* file.

### Resolving Aliases

The `sendmail` command searches the local */etc/aliases* file, or uses NIS if specified for resolving aliases. You can override the default by specifying how to resolve aliases in the */etc/netsvc.conf* file. To specify alias ordering to the `sendmail` command, enter the following:

```
alias = value [, value]
```

Use one or more of the following values for the `alias` keyword:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>files</td>
<td>Searches the local <em>/etc/aliases</em> file for the alias</td>
</tr>
<tr>
<td>nis</td>
<td>Uses NIS services for resolving alias</td>
</tr>
<tr>
<td>nis+</td>
<td>Uses NIS+ services for resolving alias</td>
</tr>
</tbody>
</table>

The order is specified on one line with values separated by commas. White spaces are permitted around the commas and the equal sign. The values specified and their ordering are dependent on the network configuration.

### Examples

1. To use only the */etc/hosts* file for resolving names, enter:

   ```
   hosts = local
   ```

2. If the resolver cannot find the name in the */etc/hosts* file and you want to the resolver to use NIS, enter:

   ```
   hosts = local , nis
   ```

3. To use the LDAP server for resolving names, indicate that it is authoritative, and to also use the BIND service, enter:

   ```
   hosts = ldap = auth , bind
   ```

   In this example, if the resolver cannot contact the LDAP server, then it searches the BIND service.

4. To override the default order and use only NIS for resolving aliases by the `sendmail` command, enter:

   ```
   aliases = nis
   ```

### Files

*/etc/netsvc.conf* Specifies the path to the file.
Related Information
The `aliases` file for mail, `irs.conf` file, `hosts` file format for TCP/IP.

The `sendmail` command.

The `gethostbyname` subroutine, `gethostbyaddr` subroutine, and `gethostent` subroutine.

TCP/IP name resolution in *Networks and communication management*.

networks File for NFS

Purpose
Contains information about networks on the NFS Internet network.

Description
The `/etc/networks` file contains information regarding the known networks that make up the Internet network. The file has an entry for each network. Each entry consists of a single line with the following information:

- Official network name
- Network number
- Aliases

Items are separated by any number of blanks or tab characters. A # (pound sign) indicates the beginning of a comment; characters up to the end of the line are not interpreted by routines that search the file.

Note: This file is not supported by the operating system. However, if this file resides on your system, Network Information Services (NIS) software will create a map for it.

Files
`/etc/networks` Specifies the path of the file.

Related Information
NFS Services in the *Networks and communication management*.

List of NFS files in the *Networks and communication management*.

NLSvec File

Purpose
Encodes PostScript fonts for the ISO8859-1 codeset characters that have code points of more than 127 decimal.

Description
The `/usr/lib/ps/NLSvec` file can contain optional comments, optional code sets, and optional character encodings.

If a line begins with an * (asterisk), it is treated as a comment.
If a specified codeset is used, it must precede all character encodings. If a code set is not specified, the default is ISO8859-1. A specified code set uses the following syntax:

\texttt{x codeset CodeSetName}

\texttt{x} Use a lowercase letter.

\texttt{codeset}

Use all lowercase letters.

\texttt{CodeSetName}

Use any valid code set name available for use with the \texttt{iconv} command.

A character encoding uses the following syntax:

\texttt{CodePoint PostscriptFontPosition PostscriptCharacterName}

\texttt{CodePoint}

Displays the decimal code point for the character.

\texttt{PostscriptFontPosition}

Displays the new encoding for that character within the PostScript fonts. The encoding can be octal or decimal.

\texttt{PostscriptCharacterName}

Displays the PostScript character name.

The PostScript assigned character encodings as well as the character names can be found in the following book:


\section*{Examples}

\section*{Notes:}

1. Following is an example of a specified codeset:

\begin{verbatim}
  x codeset ISO8859-1
\end{verbatim}

2. Following is an example of a character encoding:

\begin{verbatim}
  161 0241 exclamdown
\end{verbatim}

\section*{International Character Support}

By default, the output code set for the TranScript commands is ISO8859-1. The output code set can be specified with the \texttt{NLSvec} file. For the \texttt{enscript}, \texttt{ps4014}, \texttt{ps630}, and \texttt{psplot} TranScript commands, the input codeset is determined from the current locale. The mapping of characters outside the ASCII range is determined through the \texttt{iconv} subroutine using the input and output code sets. If there is no corresponding \texttt{iconv} converter, the commands treat the input data as if it were produced in ISO8859-1. This means that ASCII data is output correctly for all locales and codesets. For multibyte locales with no \texttt{iconv} converters to ISO8859-1 each byte of a multibyte character is treated as individual characters of the ISO8859-1 form. The only exception to this is the \texttt{enscript} command, which translates characters rather than bytes in the current locale through the mapping in the \texttt{NLSvec} file.

The following table lists the characters from the IBM-850 code set, which does not map directly to the ISO8859-1 code set through the \texttt{iconv} subroutine. The following characters would be mapped to 26 (0x1A) by the \texttt{iconv} subroutine and thus be discarded on output. It is possible to define an alternative \texttt{NLSvec} file for the IBM-850 code set so that more of the characters can be output on a PostScript device. The characters marked with an * (asterisk) before the character name are normally available in a PostScript font.
<table>
<thead>
<tr>
<th>Code Point</th>
<th>Character Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>159 (0x9F)</td>
<td>* Florin sign, PostScript name: florin</td>
</tr>
<tr>
<td>176 (0xB0)</td>
<td>Quarter hashed</td>
</tr>
<tr>
<td>177 (0xB1)</td>
<td>Half hashed</td>
</tr>
<tr>
<td>178 (0xB2)</td>
<td>Full hashed</td>
</tr>
<tr>
<td>179 (0xB3)</td>
<td>Vertical bar</td>
</tr>
<tr>
<td>180 (0xB4)</td>
<td>Right-side middle</td>
</tr>
<tr>
<td>185 (0xB9)</td>
<td>Double right-side middle</td>
</tr>
<tr>
<td>186 (0xBA)</td>
<td>Double vertical bar</td>
</tr>
<tr>
<td>187 (0xBB)</td>
<td>Double upper-right corner bar</td>
</tr>
<tr>
<td>188 (0xBC)</td>
<td>Double lower-right corner bar</td>
</tr>
<tr>
<td>191 (0xBF)</td>
<td>Upper-right corner box</td>
</tr>
<tr>
<td>192 (0xC0)</td>
<td>Lower-left corner box</td>
</tr>
<tr>
<td>193 (0xC1)</td>
<td>Bottom-side middle</td>
</tr>
<tr>
<td>194 (0xC2)</td>
<td>Top-side middle</td>
</tr>
<tr>
<td>195 (0xC3)</td>
<td>Left-side middle</td>
</tr>
<tr>
<td>196 (0xC4)</td>
<td>Center box bar</td>
</tr>
<tr>
<td>197 (0xC5)</td>
<td>Intersection</td>
</tr>
<tr>
<td>200 (0xC8)</td>
<td>Double lower-left corner bar</td>
</tr>
<tr>
<td>201 (0xC9)</td>
<td>Double upper-left corner bar</td>
</tr>
<tr>
<td>202 (0xCA)</td>
<td>Double bottom-side middle</td>
</tr>
<tr>
<td>203 (0xCB)</td>
<td>Double top-side middle</td>
</tr>
<tr>
<td>204 (0xCC)</td>
<td>Double left-side middle</td>
</tr>
<tr>
<td>205 (0xCD)</td>
<td>Double center box bar</td>
</tr>
<tr>
<td>206 (0xCE)</td>
<td>Double intersection</td>
</tr>
<tr>
<td>213 (0xD5)</td>
<td>* Small dotless i, PostScript name: dotless i</td>
</tr>
<tr>
<td>217 (0xD9)</td>
<td>Lower-right corner box</td>
</tr>
<tr>
<td>218 (0xDA)</td>
<td>Upper-left corner box</td>
</tr>
<tr>
<td>219 (0xDB)</td>
<td>Bright character cell</td>
</tr>
<tr>
<td>220 (0xDC)</td>
<td>Bright character cell lower half</td>
</tr>
<tr>
<td>223 (0xDF)</td>
<td>Bright character cell upper half</td>
</tr>
<tr>
<td>242 (0xF2)</td>
<td>Double underscore</td>
</tr>
<tr>
<td>254 (0xFE)</td>
<td>Vertical solid rectangle</td>
</tr>
</tbody>
</table>

**Files**

-XPSLIBDIRX

/usr/lib/ps/NLSvec

Specifies the /usr/lib/ps directory. This file is the default. Contains Adobe TranScript character encodings for the ISO8859-1 code set. This file is the default.

PSVECFILE

Used as an environment variable to define an NLSvec file other than the default file.

**Related Information**

The `enscript` command, `iconv` command, `ps630` command, `ps4014` command, `psplot` command.

**ntp.conf File**

**Purpose**

Controls how the Network Time Protocol (NTP) daemon `xntpd` operates and behaves.
The ntp.conf file is a basic configuration file controlling the xntpd daemon.

The following options are discussed in this article:

- Configuration Options
- Configuration Authentication Options
- Configuration Access Control Options
- Configuration Monitoring Options
- Miscellaneous Configuration Options

Configuration Options

In the ntp.conf file, comments begin with a # character and extend to the end of the line. Blank lines are ignored. Options consist of an initial keyword followed by a list of arguments, which may be optional, separated by whitespace. These options may not be continued over multiple lines. Arguments may be host names, host addresses written in numeric (dotted decimal) form, integers, floating point numbers (when specifying times in seconds) and text strings.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>peer [ HostAddress ] [ key Number ] [ version Number ] [ prefer ] [ minpoll Number ] [ maxpoll Number ]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Specifies that the local server operate in symmetric active mode with the remote server specified by HostAddress. In this mode, the local server can be synchronized to the remote server, or the remote server can be synchronized to the local server. Use this method in a network of servers where, depending on various failure scenarios, either the local or remote server host may be the better source of time.</td>
</tr>
<tr>
<td></td>
<td>The key Number specifies that all packets sent to HostAddress include authentication fields encrypted using the specified key number. The value of KeyNumber is the range of an unsigned 32 bit integer.</td>
</tr>
<tr>
<td></td>
<td>The version Number specifies the version number to use for outgoing NTP packets. The values for Version can be 1 or 2. The default is NTP version 3 implementation.</td>
</tr>
<tr>
<td></td>
<td>The prefer option marks the host as a preferred host. This host is not subject to preliminary filtering.</td>
</tr>
<tr>
<td></td>
<td>The minpoll number specifies the minimum poll interval allowed by any peer of the Internet system. The minimum poll interval is calculated, in seconds, as 2 to the power of minpoll value. The default value of minpoll is 6, i.e. the corresponding poll interval is 64 seconds.</td>
</tr>
<tr>
<td></td>
<td>The maxpoll number specifies the maximum poll interval allowed by any peer of the Internet system. The maximum poll interval is calculated, in seconds, as 2 to the power of maxpoll value. The default value of maxpoll is 10, therefore the corresponding poll interval is ~17 minutes.</td>
</tr>
<tr>
<td>server [ HostAddress ] [ key Number ] [ version Number ] [ prefer ] [ mode Number ] [ minpoll Number ] [ maxpoll Number ] [ iburst ]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The allowable range for minpoll and maxpoll is 4 (16 seconds) to 14 (~4.5 hours) inclusive.</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>Specifies that the local server operate in client mode with the remote server specified by <code>HostAddress</code>. 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.</td>
</tr>
<tr>
<td></td>
<td>The <code>key Number</code> specifies that all packets sent to <code>HostAddress</code> include authentication fields encrypted using the specified key number. The value of <code>KeyNumber</code> is the range of an unsigned 32 bit integer.</td>
</tr>
<tr>
<td></td>
<td>The <code>version Number</code> specifies the version number to use for outgoing NTP packets. The values for <code>Version</code> can be 1 or 2. The default is NTP version 3 implementation.</td>
</tr>
<tr>
<td></td>
<td>The <code>prefer</code> argument marks the host as a preferred host. This host is not subject to preliminary filtering.</td>
</tr>
<tr>
<td></td>
<td>The <code>minpoll</code> number specifies the minimum poll interval allowed by any peer of the Internet system. The minimum poll interval is calculated, in seconds, as 2 to the power of minpoll value. The default value of minpoll is 6, i.e. the corresponding poll interval is 64 seconds.</td>
</tr>
<tr>
<td></td>
<td>The <code>maxpoll</code> number specifies the maximum poll interval allowed by any peer of the Internet system. The maximum poll interval is calculated, in seconds, as 2 to the power of maxpoll value. The default value of maxpoll is 10, therefore the corresponding poll interval is ~17 minutes.</td>
</tr>
<tr>
<td>broadcast</td>
<td>Specifies that the local server operate in broadcast mode where the local server sends periodic broadcast messages to a client population at the broadcast/multicast address specified by <code>HostAddress</code>. Ordinarily, this specification applies only to the local server operating as a transmitter. In this mode, <code>HostAddress</code> is usually the broadcast address on [one of] the local network[s] or a multicast address. The address assigned to NTP is 224.0.1.1; presently, this is the only number that should be used.</td>
</tr>
<tr>
<td></td>
<td>The <code>key Number</code> specifies that all packets sent to <code>HostAddress</code> include authentication fields encrypted using the specified key number. The value of <code>Number</code> is the range of an unsigned 32 bit integer.</td>
</tr>
<tr>
<td></td>
<td>The <code>version Number</code> specifies the version number to use for outgoing NTP packets. The values for <code>Version</code> can be 1 or 2. The default is NTP version 3 implementation.</td>
</tr>
<tr>
<td></td>
<td>The <code>ttl Number</code> is used only with the broadcast mode. It specifies the time-to-live (TTL) to use on multicast packets. This value defaults to 127.</td>
</tr>
<tr>
<td></td>
<td>Specifies that the local server listen for broadcast messages on the local network in order to discover other servers on the same subnet. When the local server hears a broadcast message for the first time, it measures the nominal network delay using a brief client/server exchange with the remote server, then enters the broadcastclient mode, where it listens for and synchronizes to succeeding broadcast messages.</td>
</tr>
<tr>
<td>multICASTclient</td>
<td>Works like <code>broadcastclient</code> configuration option, but operates using IP multicasting. If you give one or more IP addresses, the server joins the respective multicast group(s). If you do not give an IP address, the IP address assumed is the one assigned to NTP (224.0.1.1).</td>
</tr>
</tbody>
</table>
Option | Description
--- | ---
**driftfile Filename** | Specifies the name of the file used to record the frequency offset of the local clock oscillator. The xntpd daemon reads this file at startup, if it exists, in order to set the initial frequency offset and then updates it once per hour with the current offset computed by the daemon. If the file does not exist or you do not give this option, the initial frequency offset assumed is zero. In this case, it may take some hours for the frequency to stabilize and the residual timing errors to subside. The file contains a single floating point value equal to the offset in parts-per-million (ppm).

**Note:** The update of the file occurs by first writing the current drift value into a temporary file and then using `rename` to replace the old version. The xntpd daemon must have write permission in the directory of the drift file, and you should avoid file system links, symbolic or otherwise.

**enable auth | bclient | pll | monitor | stats [ ... ]**

Enables various server options. Does not affect arguments not mentioned.

The **auth** option causes 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 argument is disable (off).

The **bclient** option causes the server to listen for a message from a broadcast or multicast server, following which an association is automatically instantiated for that server. The default for this argument is disable (off).

The **pll** option enables the server to adjust its local clock, with default enable (on). If not set, the local clock free-runs at its intrinsic time and frequency offset. This option is useful when the local clock is controlled by some other device or protocol and NTP is used only to provide synchronization to other clients.

The **monitor** option enables the monitoring facility, with default enable (on).

The **stats** option enables statistics facility filegen, with default enable (on).

**disable auth | bclient | pll | monitor | stats [ ... ]**

Disables various server options. Does not affect arguments not mentioned. The options are described under the **enable** subcommand.

**slewalways yes|no**

Specifies that xntpd always slews the time. The default value is **no**.

**ignore_bigtimestep yes|no**

Specifies that xntpd does not exit if no configured servers are within 1000 seconds of local system time. The default value is **no**.

**slewthreshold Seconds**

Specifies the maximum clock offset (in seconds) used for slewing. The default is 0.128 seconds.

**tracefile TraceFile**

Specifies the name of the file for debugging. (see the -o option to xntpd)

**tracelevel Number**

Specifies the debugging level. (see the -D option to xntpd)

### Configuration Authentication Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>keys Filename</strong></td>
<td>Specifies the name of a file which contains the encryption keys and key identifiers used by the xntpd daemon when operating in authenticated mode.</td>
</tr>
</tbody>
</table>
| **trustedkey Number [ Number ... ]** | Specifies the encryption key identifiers which are trusted for the purposes of authenticating peers suitable for synchronization. The authentication procedures require that both the local and remote servers share the same key and key identifier for this purpose, although you can use different keys with different servers. Each Number is a 32 bit unsigned integer.  

**Note:** The NTP key 0 is fixed and globally known. To perform meaningful authentication, the 0 key should not be trusted.
**Option**

**requestkey** *Number*

Describes the key identifier to use with the **xntpd** query/control program that diagnoses and repairs problems that affect the operation of the **xntpd** daemon. The operation of the **xntpd** query/control program is specific to this particular implementation of the **xntpd** daemon and can be expected to work only with this and previous versions of the daemon. Requests from a remote **xntpd** program which affect the state of the local server must be authenticated, which requires both the remote program and local server share a common key and key identifier. The value of *Number* is a 32 bit unsigned integer. If you do not include **requestkey** in the configuration file, or if the keys do not match, such requests are ignored.

**controlkey** *Number*

Describes the key identifier to use with the **ntpq** query program, that diagnoses problems that affect the operation of the **xntpd** daemon. The operation of the **ntpq** query program and the **xntpd** daemon conform to those specified in RFC 1305. Requests from a remote **ntpq** program which affect the state of the local server must be authenticated, which requires both the remote program and local server share a common key and key identifier. The value of *Number* is a 32 bit unsigned integer. If you do not include **controlkey** in the configuration file, or if the keys do not match, such requests are ignored.

**authdelay** *Seconds*

Specifies the amount of time it takes to encrypt an NTP authentication field on the local computer. This value corrects transmit timestamps when using authentication on outgoing packets. The value usually lies somewhere in the range 0.0001 seconds to 0.003 seconds, though it is very dependent on the CPU speed of the host computer.

---

**Configuration Access Control Options**

The **xntpd** daemon inserts default restriction list entries, with the parameters **ignore** and **ntpport**, for each of the local host’s interface addresses into the table at startup to prevent the server from attempting to synchronize to its own time. A default entry is also always present, though if it is otherwise unconfigured it does not associate parameters with the default entry (everything besides your own NTP server is unrestricted).

While this facility may be useful for keeping unwanted or broken remote time servers from affecting your own, do not consider it an alternative to the standard NTP authentication facility.

**restrict** *Address [ mask Number | default ] [ Parameter ... ]*

Specifies the restrictions to use on the given address. The **xntpd** daemon implements a general purpose address-and-mask based restriction list. The **xntpd** daemon sorts this list by address and by mask, and searches the list in this order for matches, with the last match found defining the restriction flags associated with the incoming packets. The **xntpd** daemon uses the source address of incoming packets for the match, doing a logical and operation with the 32 bit address and the mask associated with the restriction entry. It then compares it with the entry’s address (which has also been and’ed with the mask) to look for a match. The **mask** option defaults to 255.255.255.255, meaning that *Address* is treated as the address of an individual host. A default entry (address 0.0.0.0, mask 0.0.0.0) is always included and is always the first entry in the list. The text string **default**, with no mask option, may be used to indicate the default entry.

In the current implementation, **Parameter** always restricts access. An entry with no **Parameter** gives free access to the server. More restrictive **Parameters** will often make less restrictive ones redundant. The **Parameters** generally restrict time service or restrict informational queries and attempts to do run time reconfiguration of the server. You can specify one or more of the following value for **Parameter**:

**ignore**

Specifies to ignore all packets from hosts which match this entry. Does not respond to queries nor time server polls.
limited
Specifies that these hosts are subject to limitation of number of clients from the same net. Net in this context refers to the IP notion of net (class A, class B, class C, and so on). Only accepts the first client_limit hosts that have shown up at the server and that have been active during the last client_limit_period seconds. Rejects requests from other clients from the same net. Only takes into account time request packets. Private, control, and broadcast packets are not subject to client limitation and therefore do not contribute to client count. The monitoring capability of the xntpd daemon keeps a history of clients. When you use this option, monitoring remains active. The default value for client_limit is 3. The default value for client_limit_period is 3600 seconds.

lowpriotrap
Specifies to declare traps set by matching hosts to low-priority status. The server can maintain a limited number of traps (the current limit is 3), assigned on a first come, first served basis, and denies service to later trap requestors. This parameter modifies the assignment algorithm by allowing later requests for normal priority traps to override low-priority traps.

nomodify
Specifies to ignore all NTP mode 6 and 7 packets which attempt to modify the state of the server (run time reconfiguration). Permits queries which return information.

nopeer
Specifies to provide stateless time service to polling hosts, but not to allocate peer memory resources to these hosts.

noquery
Specifies to ignore all NTP mode 6 and 7 packets (information queries and configuration requests) from the source. Does not affect time service.

noserve
Specifies to ignore NTP packets whose mode is not 6 or 7. This denies time service, but permits queries.

notrap
Specifies to decline to provide mode 6 control message trap service to matching hosts. The trap service is a subsystem of the mode 6 control message protocol intended for use by remote event-logging programs.

notrust
Specifies to treat these hosts normally in other respects, but never use them as synchronization sources.

ntpport
Specifies to match the restriction entry only if the source port in the packet is the standard NTP UDP port (123).

clientlimit Number
Sets client_limit. Specifies the number of clients from the same network allowed to use the server. Allows the configuration of client limitation policy.

clientperiod Seconds
Sets client_limit_period. Specifies the number of seconds to before considering if a client is inactive and no longer counted for client limit restriction. Allows the configuration of client limitation policy.

Configuration Monitoring Options
File generation sets manage statistical files. The information obtained by enabling statistical recording allows analysis of temporal properties of a server running the xntpd daemon. It is usually only useful to primary servers.
statsdir DirectoryPath
Specifies the full path of the directory in which to create statistical files. Allows modification of the otherwise constant filegen filename prefix for file generation sets used for handling statistical logs.

statistics Type...
Enables writing of statistical records. The following are the types of statistics supported:

loopstats
Enables recording of loop filter statistical information. Each update of the local clock outputs a line of the following format to the file generation set named loopstats:

```
48773 10847.650 0.0001307 17.3478 2
```

The first two fields show the date (Modified Julian Day) and time (seconds and fraction past UTC midnight). The next three fields show time offset in seconds, frequency offset in parts-per-million and time constant of the clock-discipline algorithm at each update of the clock.

peerstats
Enables recording of peer statistical information. This includes statistical records of all peers of an NTP server and of the 1-pps signal, where present and configured. Each valid update appends a line of the following format to the current element of a file generation set named peerstats:

```
48773 10847.650 127.127.4.1 9714 -0.001605 0.00000 0.00142
```

The first two fields show the date (Modified Julian Day) and time (seconds and fraction past UTC midnight). The next two fields show the peer address in dotted-quad notation and status, respectively. The status field is encoded in hex in the format described in Appendix A of the NTP specification RFC 1305. The final three fields show the offset, delay and dispersion, all in seconds.

clockstats
Enables recording of clock driver statistical information. Each update received from a clock driver outputs a line of the following form to the file generation set named clockstats:

```
49213 525.624 127.127.4.1 93 226
```

The first two fields show the date (Modified Julian Day) and time (seconds and fraction past UTC midnight). The next field shows the clock address in dotted-quad notation, The final field shows the last timecode received from the clock in decoded ASCII format, where meaningful. You can gather and display a good deal of additional information in some clock drivers.

filegen Name [ file FileName ] [ type TypeName ] [ flag flagval ] [ link ] [ nolink ] [ enable ] [ disabled ]
Configures setting of generation fileset name. Generation filesets provide a means for handling files that are continuously growing during the lifetime of a server. Server statistics are a typical example for such files. Generation filesets provide access to a set of files used to store the actual data. A file generation set is characterized by its type. At any time, at most one element of the set is being written to. Filenames of set members are built from three elements:

Prefix This is a constant filename path. It is not subject to modifications with the filegen option. It is defined by the server, usually specified as a compile time constant. You can, however, configure it for individual file generation sets with other commands. For example, you can configure the prefix used with loopstats and peerstats filegens using the statsdir option.

file FileName
The string FileName is directly concatenated to the prefix with no intervening slash (/). You
can modify this by using the file argument to the filegen option. To prevent filenames referring to parts outside the filesystem hierarchy denoted by prefix, ".." elements are not allowed in this component

**Suffix**
This part reflects individual elements of a fileset. It is generated according to the type of a fileset.

**type** *TypeName*
Specifies when and how to direct data to a new element of the set. This way, information stored in elements of a fileset that are currently unused are available for administrational operations without the risk of disturbing the operation of the xntpd daemon. Most important, you can remove them to free space for new data produced. The following types are supported:

- **none**
  Specifies that the fileset is actually a single plain file.

- **pid**
  Specifies the use of one element of fileset per server running the xntpd daemon. This type does not perform any changes to fileset members during runtime; however, it provides an easy way of separating files belonging to different servers running the xntpd daemon. The set member filename is built by appending a dot (.) to concatenated prefix and strings denoted in file Name, and appending the decimal representation of the process id of the xntpd server process.

- **day**
  Specifies the creation of one file generation set element per day. The term day is based on UTC. A day is the period between 00:00 and 24:00 UTC. The fileset member suffix consists of a dot (.) and a day specification in the form YYYYMMDD, where YYYY is a 4 digit year number, MM is a two digit month number, and, DD is a two digit day number. For example, all information written at January 10th, 1992 would end up in a file named PrefixFileName.19920110.

- **week**
  Specifies the creation of one file generation set element per week. A week is computed as day-of-year modulo 7. The fileset member suffix consists of a dot (.), a four digit year number, the letter W, and a two digit week number. For example, all information written at January, 10th 1992 would end up in a file named PrefixFileName.1992W1.

- **month**
  Specifies the creation of one file generation set element per month. The fileset member suffix consists of a dot (.), a four digit year number, and a two digit month number. For example, all information written at January, 1992 would end up in a file named PrefixFileName.199201.

- **year**
  Specifies the creation of one file generation set element per year. The fileset member suffix consists of a dot (.) and a four digit year number. For example, all information written at January, 1992 would end up in a file named PrefixFileName.1992.

- **age**
  Specifies the creation of one file generation set element every 24 hours of server operation. The fileset member suffix consists of a dot (.), the letter a, and an eight digit number. This number is the number of seconds of run-time of the server since the start of the corresponding 24 hour period.

**enable**
Enables the writing of information to a file generation set.

**disabled**
Disables the writing of information to a file generation set.

**link**
Enables the access of the current element of a file generation set by a fixed name by creating a hard link from the current fileset element to a file without Suffix. If a file with this name already exists and the number of links of this file is one, it is renamed by appending a dot (.), the letter C,
and the pid of the `xntpd` server process. If the number of links is greater than one, the file is unlinked. This allows access of the current file by a constant name.

**nolink** Disables access the current element of a file generation set by a fixed name.

### Miscellaneous Configuration Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>precision</td>
<td>Specifies the nominal precision of the local clock. The <code>Number</code> is an integer approximately equal to the base 2 logarithm of the local timekeeping precision in seconds. Normally, the <code>xntpd</code> daemon determines the precision automatically at startup, so use this option when the <code>xntpd</code> daemon cannot determine the precision automatically.</td>
</tr>
<tr>
<td>broadcastdelay</td>
<td>Specifies the default delay to use when in broadcast or multicast modes. These modes require a special calibration to determine the network delay between the local and remote servers. Normally, this is done automatically by the initial protocol exchanges between the local and remote servers. In some cases, the calibration procedure may fail due to network or server access controls, for example. These modes require a special calibration to determine the network delay between the local and remote servers. Normally, this is done automatically by the initial protocol exchanges between the local and remote servers. In some cases, the calibration procedure may fail due to network or server access controls, for example.</td>
</tr>
<tr>
<td>trap</td>
<td>Configures a trap receiver at the given host address and port number for sending messages with the specified local interface address. If you do not specify the port number, the value defaults to 18447. If you do not specify the interface address, the value defaults to the source address of the local interface. Note: On a multihomed host, the interface used may vary from time to time with routing changes. Normally, the trap receiver logs event messages and other information from the server in a log file. While such monitor programs may also request their own trap dynamically, configuring a trap receiver ensures that when the server starts, no messages are lost.</td>
</tr>
</tbody>
</table>
| setvar        | Specifies to add an additional system variable. You can use these variables to distribute additional information such as the access policy. If `default` follows a variable of the from `Name=Value`, then the variable becomes part of the default system variables, as if you used the `ntpq rv` command. These additional variables serve informational purposes only; they are not related to the protocol variables. The known protocol variables always override any variables defined with `setvar`. There are three special variables that contain the names of all variables of the same group. The `sys_var_list` holds the names of all system variables, the `peer_var_list` holds the names of all peer variables, and the `clock_var_list` holds the names of the reference clock variables.

Chapter 1. System Files 163
Option

logconfig Key

Description
Controls the amount of output written to syslog or the logfile. By default all output is turned on. You can prefix all KeyWords with = (equal), + (plus) and - (dash). You can control four classes of messages: sys, peer, clock, and sync. Within these classes, you can control four types of messages:

info Outputs informational messages that control configuration information.

events Outputs event messages that control logging of events (reachability, synchronization, alarm conditions).

status Outputs statistical messages that describe mainly the synchronization status.

all Outputs all messages having to do with the specified class and suppresses all other events and messages of the classes not specified.

You form the KeyWord by concatenating the message class with the event class. To just list the synchronization state of xntp and the major system events, enter:

logconfig =syncstatus +sysevents

To list all clock information and synchronization information and have all other events and messages about peers, system events and so on suppressed, enter:

logconfig =syncall +clockall

Files

/etc/ntp.conf Specifies the path to the file.

Related Information
The xntpd command, the xntpd daemon.

The ntp.keys file.

ntp.keys File

Purpose
Contains key identifiers and keys controlling authentication of Network Time Protocol (NTP) transactions.

Description
The ntp.keys file contains key identifiers and keys for encryption and decryption of authentication of NTP transactions.

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. The xntpd daemon implements this by using the MD5 algorithm to compute a message-digest. The specification allows any one of possibly 4 billion keys, numbered with 32 bit key identifiers, to be used to authenticate an association. The servers involved in an association must agree on the key and key identifier used to authenticate their data, although they must each learn the key and key identifier independently.
The \texttt{ntpd} daemon reads its keys from a file specified with the \texttt{-k} flag or the \texttt{keys} statement in the
configuration file. You cannot change key number 0 because the NTP standard fixes it as 64 zero bits.

The \texttt{ntp.keys} file uses the same comment conventions as the configuration file, \texttt{ntp.conf}. Key entries use
the following format:

\begin{verbatim}
KeyNumber M Key
\end{verbatim}

where,

\begin{description}
\item[Entry] \texttt{KeyNumber}
\item[Description] A positive integer
\item[] \texttt{M}
\item[Description] Specifies that \texttt{Key} is a 1-to-8 character ASCII string, using the MD5 authentication scheme.
\item[] \texttt{Key}
\item[Description] The key itself.
\end{description}

One of the keys may be chosen, by way of the \texttt{ntp.conf} configuration file \texttt{requestkey} statement, to
authenticate run-time configuration requests made using the \texttt{xntpdc} command. The \texttt{xntpdc} command
obtains the key from the terminal as a password, so it is generally appropriate to specify the key in ASCII format.

\section*{Files}

\texttt{/etc/ntp.keys} Specifies the path to the file.

\section*{Related Information}

The \texttt{xntpdc} command, the \texttt{ntpd} daemon.

\section*{objects File}

\section*{Purpose}

Contains the audit events for audited objects (files).

\section*{Description}

The \texttt{/etc/security/audit/objects} file is an ASCII stanza file that contains information about audited objects
(files). This file contains one stanza for each audited file. The stanza has a name equal to the path name
of the file.

Each file attribute has the following format:

\begin{verbatim}
access_mode = "audit_event"
\end{verbatim}

An audit-event name can be up to 15 bytes long; longer names are rejected. Valid access modes are read
(\texttt{r}), write (\texttt{w}), and execute (\texttt{x}) modes. For directories, search mode is substituted for execute mode.

\section*{Security}

Access Control: This file should grant read (\texttt{r}) access to the root user and members of the audit group and
grant write (\texttt{w}) access only to the root user.

\section*{Examples}

To define the audit events for the \texttt{/etc/security/passwd} file, add a stanza to the \texttt{/etc/security/audit/}
\texttt{objects} file. For example:
These attributes generate a S_PASSWD_READ audit event each time the passwd file is read, and a S_PASSWD_WRITE audit event each time the file is opened for writing.

Files

/etc/security/audit/objects
    Specifies the path to the file.
/etc/security/audit/config
    Contains audit system configuration information.
/etc/security/audit/events
    Contains the audit events of the system.
/etc/security/audit/bincmds
    Contains auditbin backend commands.
/etc/security/audit/streamcmds
    Contains auditstream commands.

Related Information

The audit command.
The auditobj subroutine.

Setting Up Auditing in Operating system and device management.
Auditing overview Security Administration in Operating system and device management.

pam_aix Module

Purpose
Provides AIX style authentication, account management, password management, and session management for PAM.

Description
The pam_aix module provides AIX style authentication behaviors to PAM. The module has support for each of the PAM module types - authentication, account management, password management and session management. Each of these types provides full AIX support for users defined in local or remote registries.

Communication from the pam_aix module to the user is handled through the PAM_CONV item, which is set by pam_start or pam_set_item. All messages displayed by pam_aix are AIX messages and are internationalized.

Typical usage for the pam_aix module is to be used as a backup, or “other” service. This way if a specific authentication stack is not defined for a service, local AIX authentication is used. pam_aix should usually be a “required” or “requisite” module. If used for password authentication, pam_aix should be marked as being “required” or “requisite”.

```
# Use AIX system authentication
#OTHER auth  required /usr/lib/security/pam_aix
OTHER account required /usr/lib/security/pam_aix
OTHER session required /usr/lib/security/pam_aix
OTHER password required /usr/lib/security/pam_aix
```
Attention: The pam_aix module cannot be used with a user who has their SYSTEM or registry user attribute set to use the /usr/lib/security/PAM module. This will create an authentication loop and the operation will fail.

Supported PAM module types

Authentication
Authenticates a user through their AIX password.

Account Management
Verifies that an authenticated user is permitted onto the system and checks for expired passwords. Checks are performed through use of the passwdexpired() and loginrestrictions() subroutines.

Session Management
Opens a new session and logs the session information.

Password Management
Allows a user to set or modify their AIX password if it is possible. pam_aix will then update the user’s password entry in the appropriate password table. When pam_aix is used for password management, it should be used as "required" or "requisite".

Options: The pam_aix module accepts the following parameters specified as options in the PAM configuration file:

- **debug**: Log debugging information to syslog.
- **mode**: Specifying the mode option for a service allows the login restrictions checks to be customized as needed for a PAM service. The value specified for mode can be one of the following strings:
  - S_DAEMON
  - S_LOGIN
  - S_RLOGIN
  - S_SU
  - S_DIST_CLNT
  - S_DIST_SERV

  The checks performed by each mode are defined in the loginrestrictions subroutine man page. If the option is not specified, then a mode of 0 is passed into the subsequent loginrestrictions invocation. This option is only valid for the Authentication and Account Management module types.

- **nowarn**: Do not display warning messages.
- **use_first_pass**: Use a previously entered password, do not prompt for a new one.
- **try_first_pass**: Try a previously entered password. If it fails, prompt for a new one.
- **use_new_state**: AIX builds and maintains state information when authenticating a user. By default, the pam_aix module will use the same state information throughout a PAM session. This can produce results that are correct in terms of AIX authentication but are unexpected within the PAM framework. For example, pam_authenticate requests may fail due to access restrictions. If this behavior is not desired for a given module type, specify the use_new_state option to use new state information for each invocation.

Return Values: Upon successful completion the pam_aix module returns PAM_SUCCESS. If a failure occurs a PAM error code will be returned, depending on the actual error.

Location
/usr/lib/security/pam_aix
Related Information

The loginrestrictions Subroutine, pam_acct_mgmt Subroutine, pam_authenticate Subroutine, pam_chauthtok Subroutine, pam_open_session Subroutine, pam_close_session Subroutine, pam_setcred Subroutine, and passwdexpired Subroutine.

The "pam.conf File" on page 174.


---

**pam_allow Module**

**Purpose**

Returns PAM_SUCCESS for all PAM module types.

**Description**

The **pam_allow** module returns PAM_SUCCESS to all calling applications unless an invalid PAM handle is specified. Support for all four module types is provided.

**Attention:** This module should be used with caution and should often only be used for PAM debugging purposes. Placing this module in the PAM stack for a service could potentially grant access to all users.

Functionality opposite to that provided by **pam_allow** can be obtained through use of the **pam_prohibit** module.

**Supported PAM module types**

**Authentication**

Returns PAM_SUCCESS if valid PAM handle.

**Account Management**

Returns PAM_SUCCESS if valid PAM handle.

**Session Management**

Returns PAM_SUCCESS if valid PAM handle.

**Password Management**

Returns PAM_SUCCESS if valid PAM handle.

**Options:** The **pam_allow** module accepts the following parameters specified as options in the PAM configuration file:

- `debug` Log debugging information to syslog.
- `nowarn` Do not display warning messages.

**Return Values:** Upon successful completion the **pam_allow** module returns PAM_SUCCESS. If an invalid PAM handle was specified, PAM_SYSTEM_ERR is returned.

**Location**

/usr/lib/security/pam_allow

**Related Information**

The **pam_authenticate Subroutine**, **pam_setcred Subroutine**, **pam_chauthtok Subroutine**, **pam_open_session Subroutine**, **pam_close_session Subroutine**, **pam_acct_mgmt Subroutine**, **passwdexpired Subroutine**, and **loginrestrictions Subroutine**.
pam_allowroot Module

Purpose
Returns PAM_SUCCESS if the authenticating user has a real user ID (UID) of 0.

Description
The pam_allowroot module checks the real user ID (UID) under which the PAM application was run. If the UID of the authenticating user is 0 (zero), then it is the root user and PAM_SUCCESS is returned.

The pam_allowroot module only checks the real user ID. Many applications that require root access will set the effective user ID to 0. For this reason, the effective ID is not used in determining whether or not the user executing the authenticating application is a root user.

It is recommended that pam_allowroot be used as sufficient in conjunction with other modules. This allows the root user to bypass the rest of the modules in the stack and for a failure not to impact the result of other authenticating users. An example authentication stack configuration is shown below which mimics the historic behavior of the su command.

```
# The PAM configuration for standard su behavior.
# su auth sufficient /usr/lib/security/pam_allowroot
# su auth required /usr/lib/security/pam_aix
```

Supported PAM module types

Authentication
Returns PAM_SUCCESS if UID of authenticating user is 0.

Account Management
Returns PAM_SUCCESS if UID of authenticating user is 0.

Options: The pam_allowroot module accepts the following parameters specified as options in the PAM configuration file:

- `debug`: Log debugging information to syslog.
- `nowarn`: Do not display warning messages.

Return Values: Upon successful completion, PAM_SUCCESS is returned. If a failure occurs, a PAM error code will be returned, depending on the actual error.

Location
/usr/lib/security/pam_allowroot

Related Information
The pamAuthenticate Subroutine, pamSetcred Subroutine

pam_ckfile Module

Purpose
Denies all non-root user logins if /etc/nologin or an optionally specified file is present.

Description
The pam_ckfile module allows or denies authentication, based on the existence of a file. The file checked for existence can be set with the file=<filename> module option. If not specified the file defaults to /etc/nologin.

If the specified file exists, only root users (those with a user ID of 0) may authenticate. All other users are denied access for the service, and pam_ckfile will echo the contents (if any) of that file. If the specified file does not exist, the module returns PAM_IGNORE. System administrators should ensure that success or failure of the module stack for a service does not depend solely on the result of this module.

It is recommended that pam_ckfile is used as "required" or "requisite" in conjunction with other modules. An example authentication stack is provided below to demonstrate how /etc/nologin behavior with the login service can be implemented.

```
# Provide the standard /etc/nologin behavior for login.
login auth required /usr/lib/security/pam_ckfile file=/etc/nologin
login auth required /usr/lib/security/pam_aix
```

Supported PAM module types

Authentication
Denies non-root user authentication if the specified file is present.

Account Management
Denies account access for non-root users if the specified file is present.

Options: The pam_allowroot module accepts the following parameters specified as options in the PAM configuration file:

- **debug**
  Log debugging information to syslog.

- **nowarn**
  Do not display warning messages.

- **file=<filename>**
  Use <filename> instead of /etc/nologin.

  **Note:** <filename> must be the full path to the file.

Return Values: Upon successful completion PAM_SUCCESS is returned. If the specified file does not exist the module returns PAM_IGNORE. If another failure occurs, a PAM error code will be returned, depending on the actual error.

Location
/usr/lib/security/pam_ckfile

Related Information

The "pam.conf File" on page 174.

pam_permission Module

Purpose

Allows or prohibits authentication through a configurable file containing a list of users and/or groups.

Description

The pam_permission module is an authentication and account-service PAM module that uses an access-control list to determine whether or not to permit or deny authentication requests. The file to use for the control list is configured via a module option and defaults to /etc/ftpusers if not specified.

If the access-control file exists, the pam_permission module will scan the file using the authenticating user name and group(s). The first match will then be used to determine the result. The general syntax for an entry in the access-control file is as follows:

```
[+|-]@[<name>]
```

The optional first character controls whether to allow(+) or deny(-) the request for the user or group specified by <name>. If a '+' or '-' is not the first character in an entry, then the value of the found=<action> module option determines the behavior.

Preceding a name by the '@' symbol designates the entry as a group. Otherwise the entry is used as a user name. The first match found to a user name or group entry is used to determine access.

All spaces in an entry are ignored. Comments may be added to the file using the '#' character as the first character in the line. Only one entry or comment is allowed per line and the entries are processed one at a time, sequentially, starting at the top of the file.

Using the keyword "ALL" for <name> will match all users. Since the file is parsed sequentially, use of the "ALL" keyword should be reserved for the end of the file as any entries after it are ignored.

Upon reaching the end of the access-control file, if a match to a user name or group has not been made, the result will be the opposite value of the found=<action> module option. For example, if found=prohibit is set and the user is not found within the file, then the result for that user would be allow.

If the specified access control file does not exist, the module will return PAM_IGNORE and have no affect on the module stack. It is not recommended that the overall success or failure of the module stack depend solely on pam_permission.

It is recommended that pam_permission is used as "required" or "requisite" in conjunction with other modules. An example authentication stack is provided below to demonstrate how /etc/ftpusers behavior with the ftp service can be implemented.

```
# # Provide /etc/ftpusers access-control
# to PAM-enabled ftp.
#
ftp auth requisite /usr/lib/security/pam_permission
   file=/etc/ftpusers found=prohibit
ftp auth required /usr/lib/security/pam_aix
```

Supported PAM module types

Authentication

Provides user authentication based on the contents of the access-control file.

Account Management

Provides account access and denial based upon the rules in the access-control file.
Options: The pam_permission module accepts the following parameters specified as options in the PAM configuration file:

- **debug**
  - Log debugging information to syslog.
- **nowarn**
  - Do not display warning messages.
- **file=<<filename>>**
  - Use <<filename>> as access control file. Defaults to /etc/ftpusers.
- **found={allow | prohibit}**
  - Action if an entry match was found but is not preceded by a '+' or '-' . Default is prohibit.

Return Values: Upon successful completion PAM_SUCCESS is returned. If a failure occurs, a PAM error code will be returned, depending on the actual error.

Location
/usr/lib/security/pam_permission

Related Information
The [pam_authenticate Subroutine](#) and [pam_setcred Subroutine](#).

The “pam.conf File” on page 174.


---

### pam_prohibit Module

**Purpose**

Returns a value denoting a failure for each PAM module type.

**Description**

The pam_prohibit module returns a failure for all PAM module types. If used as a required or requisite module for a service, the stack that this module is incorporated into will always fail. It is recommended that individual services be explicitly configured in /etc/pam.conf and then the pam_prohibit module used for the OTHER service entries. Configuring the system in this way ensures that only known PAM enabled applications are capable of successfully authenticating users. Listed below is an example of how to configure the OTHER service keyword in /etc/pam.conf to use the pam_prohibit module:

```
# Fail for all PAM services not explicitly configured
#
OTHER auth required /usr/lib/security/pam_prohibit
OTHER account required /usr/lib/security/pam_prohibit
OTHER password required /usr/lib/security/pam_prohibit
OTHER session required /usr/lib/security/pam_prohibit
```

Functionality opposite to that provided by pam_prohibit can be obtained by using the pam_allow module.

**Supported PAM module types**

**Authentication**
- **pam_sm_authenticate** returns PAM_AUTH_ERR
- **pam_sm_setcred** returns PAM_CRED_ERR

**Account Management**
- **pam_sm_acct_mgmt** returns PAM_ACCT_EXPIRED
Session Management
   pam_sm_open_session returns PAM_SESSION_ERR

Password Management
   pam_sm_chauthtok returns PAM_AUTHTOK_ERR

Options: The pam_prohibit module accepts the following parameters specified as options in the PAM configuration file:

   debug   Log debugging information to syslog.
   nowarn  Do not display warning messages.

Return Values: The pam_prohibit module will never return PAM_SUCCESS. If an invalid PAM handle is found then PAM_SYSTEM_ERR is returned, otherwise the error code returned is PAM module type specific.

Location
/usr/lib/security/pam_prohibit

Related Information
The pam_acct_mgmt Subroutine, pam_authenticate Subroutine, pam_chauthtok Subroutine, pam_close_session Subroutine, pam_open_session Subroutine, pam_setcred Subroutine.

The "pam.conf File" on page 174.


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

pam_rhosts_auth Module

Purpose
Provides rhosts-based authentication for PAM.

Description
The pam_rhosts_auth module provides rhost authentication services similar to the rlogin, rsh, and rcp commands. The module queries the PAM handle for the remote user name, remote host, and the local user name. This information is then compared to the rules in /etc/hosts.equiv and $HOME/.rhosts.

For a typical user, the module first checks /etc/hosts.equiv. If a match is not found for the username and hostname, the module will continue on to check the $HOME/.rhosts file. If a username and hostname match is still not found, the module returns the PAM_AUTH_ERR failure code. Otherwise, the result depends on the first rule found matching the specified username and hostname.

When authenticating to the root user (user with the UID of 0), the first check of the /etc/hosts.equiv file is skipped. Success of the rhosts authentication is based solely on the contents of the root user’s $HOME/.rhosts file.

This module requires that a PAM application, before making the call to pam_authenticate, call pam_set_item and at least set the values of PAM_RHOST and PAM_RUSER. If the PAM_USER item is not set, the module will prompt for the user name through the conversation function provided in the PAM handle.
Further description on how rhosts authentication works can be found in the documentation for the ruserok() subroutine. Information regarding the syntax of rhost configuration files can be found in the $HOME/.rhosts or /etc/hosts.equiv files description.

For expected behavior, pam_rhosts_auth should be used as one of the first authentication modules in the stack and designated as sufficient.

```
# # PAM authentication stack for typical rlogin behavior.
#
rlogin auth sufficient /usr/lib/security/pam_rhosts_auth
rlogin auth required /usr/lib/security/pam_aix
```

Supported PAM module types

Authentication

Authenticates a user through rhost-based authentication.

Options: The pam_rhosts_auth module accepts the following parameters specified as options in the PAM configuration file:

- **debug**: Log debugging information to syslog.
- **nowarn**: Do not display warning messages.

Return Values: Upon successful completion PAM_SUCCESS is returned. If a failure occurs, a PAM error code will be returned, depending on the actual error.

Location

/usr/lib/security/pam_rhosts_auth

Related Information

The pam_authenticate Subroutine, pam_setcred Subroutine, pam_set_item Subroutine, ruserok Subroutine.


The rlogin Command, rsh Command and rcp Command


pam.conf File

Purpose

Contains service entries for each PAM (Pluggable Authentication Modules) module type.

Description

The /etc/pam.conf configuration file consists of service entries for each PAM module type and serves to route services through a defined module path. Entries in the file are composed of the following whitespace-delimited fields: service_name module_type control_flag module_path module_options

<table>
<thead>
<tr>
<th>service_name</th>
<th>Specifies the name of the PAM enabled service. The keyword OTHER is used to define the default module to use for applications not specified in an entry.</th>
</tr>
</thead>
</table>
module_type
Specifies the module type for the service. Valid module types are auth, account, session, or password. A given module will provide support for one or more module types.

control_flag
Specifies the stacking behavior for the module. Supported control flags are required, requisite, sufficient, or optional.

required
All required modules in a stack must pass for a successful result. If one or more of the required modules fail, all of the required modules in the stack will be attempted, but the error from the first failed required module is returned.

requisite
Similar to required except that if a requisite module fails, no further modules in the stack are processed and it immediately returns the first failure code from a required or requisite module.

sufficient
If a module flagged as sufficient succeeds and no previous required or sufficient modules have failed, all remaining modules in the stack are ignored and success is returned.

optional
If none of the modules in the stack are required and no sufficient modules have succeeded, then at least one optional module for the service must succeed. If another module in the stack is successful, a failure in an optional module is ignored.

module_path
Specifies the module to load for the service. Valid values for module_path may be specified as either the full path to the module or just the module name. If the full path to the module is not specified, the PAM library prepends /usr/lib/security (for 32-bit services) or /usr/lib/security/64 (for 64-bit services) to the module name.

module_options
Specifies a space delimited list of module specific options. Values for this field are dependent on the options supported by the module defined in the module_path field. This field is optional.

Malformed entries, or entries with invalid values for the module_type or control_flag fields are ignored by the PAM library. Entries beginning with a number sign (#) character at the beginning of the line are also ignored as this denotes a comment.

PAM supports a concept typically referred to as stacking, which allows multiple mechanisms to be used for each service. Stacking is implemented in the configuration file by creating multiple entries for a service with the same module_type field. The modules are invoked in the order in which they are listed in the file for a given service name, with the final result determined by the control_flag field specified for each entry.

The following /etc/pam.conf subset is an example of stacking in the auth module type for the login service.

```
# Authentication Management
login auth required /usr/lib/security/pam_ckfile file=/etc/nologin
login auth required /usr/lib/security/pam_aix
login auth optional /usr/lib/security/pam_test use_first_pass
OTHER auth required /usr/lib/security/pam_prohibit
```

The example configuration file contains three entries for the login service. Having specified both pam_ckfile and pam_aix as required, both modules will be executed and both must be successful for the overall result to be success. The third entry for the fictitious pam_test module is optional and its success or failure will not affect whether the user is able to login. The option use_first_pass to the pam_test module requires that a previously entered password be used instead of prompting for a new one.

Chapter 1. System Files 175
Use of the OTHER keyword as a service name enables a default to be set for any other services that are not explicitly declared in the configuration file. Setting up a default ensures that all cases for a given module type will be covered by at least one module. In the case of this example, all services other than login will always fail since the pam_prohibit module returns a PAM failure for all invocations.

### Changing the `/etc/pam.conf` File

When changing the `/etc/pam.conf` configuration file, consider the following:

- The file should always be owned by the root user and group security. Permission on the file should be set to 644 to allow everyone read access, but only allow root to modify it.
- For greater security, consider explicitly configuring each PAM enabled service and then using the `pam_prohibit` module for the OTHER service keyword.
- Read any documentation supplied for a chosen module and service, and determine which control flags, options and module types are supported and what their impact will be.
- Select the ordering of modules and control flags carefully, keeping in mind the behavior of required, requisite, sufficient, and optional control flags in stacked modules.

**Note:** Incorrect configuration of the PAM configuration file can result in a system that cannot be logged in to since the configuration applies to all users, including root. After making changes to the file, always test the affected applications before logging out of the system. A system that cannot be logged in to can be recovered by booting the system in maintenance mode and correcting the `/etc/pam.conf` configuration file.

### Files

- **/etc/pam.conf** Location of the `pam.conf` configuration file.

### `/etc/passwd` File

#### Purpose

Contains basic user attributes.

#### Description

The `/etc/passwd` file contains basic user attributes. This is an ASCII file that contains an entry for each user. Each entry defines the basic attributes applied to a user. When you use the `mkuser` command to add a user to your system, the command updates the `/etc/passwd` file.

**Note:** Certain system-defined group and user names are required for proper installation and update of the system software. Use care before replacing this file to ensure that no system-supplied groups or users are removed.

An entry in the `/etc/passwd` file has the following form:

**Name:**Password:*UserID:*PrincipleGroup:*Gecos:*HomeDirectory:*Shell

Attributes in an entry are separated by a : (colon). For this reason, you should not use a : (colon) in any attribute. The attributes are defined as follows:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Specifies the user’s login name. There are a number of restrictions on naming users. See the <code>mkuser</code> command for more information.</td>
</tr>
</tbody>
</table>
### Attribute Definition

**Password**
Contains an * (asterisk) indicating an invalid password or an ! (exclamation point) indicating that the password is in the `/etc/security/passwd` file. Under normal conditions, the field contains an !. If the field has an * and a password is required for user authentication, the user cannot log in.

**UserID**
Specifies the user's unique numeric ID. This ID is used for discretionary access control. The value is a unique decimal integer.

**PrincipleGroup**
Specifies the user's principal group ID. This must be the numeric ID of a group in the user database or a group defined by a network information service. The value is a unique decimal integer.

**Gecos**
Specifies general information about the user that is not needed by the system, such as an office or phone number. The value is a character string. The Gecos field cannot contain a colon.

**HomeDirectory**
Specifies the full path name of the user's home directory. If the user does not have a defined home directory, the home directory of the guest user is used. The value is a character string.

**Shell**
Specifies the initial program or shell that is executed after a user invokes the `login` command or the `su` command. If a user does not have a defined shell, `/usr/bin/sh`, the system shell, is used. The value is a character string that may contain arguments to pass to the initial program.

Users can have additional attributes in other system files. See the ‘Files’ section for additional information.

### Changing the User File

You should access the user database files through the system commands and subroutines defined for this purpose. Access through other commands or subroutines may not be supported in future releases. Use the following commands to access user database files:

- `chfn`
- `chsh`
- `chuser`
- `lsuser`
- `mkuser`
- `rmuser`

The `mkuser` command adds new entries to the `/etc/passwd` file and fills in the attribute values as defined in the `/usr/lib/security/mkuser.default` file.

The `Password` attribute is always initialized to an * (asterisk), an invalid password. You can set the password with the `passwd` or `pwdadm` command. When the password is changed, an ! (exclamation point) is added to the `/etc/passwd` file, indicating that the encrypted password is in the `/etc/security/passwd` file.

Use the `chuser` command to change all user attributes except Password. The `chfn` command and the `chsh` command change the Gecos attribute and Shell attribute, respectively. To display all the attributes in this file, use the `lsuser` command. To remove a user and all the user's attributes, use the `rmuser` command.

To write programs that affect attributes in the `/etc/passwd` file, use the subroutines listed in Related Information.

### Security

Access Control: This file should grant read (r) access to all users and write (w) access only to the root user and members of the security group.
Examples

1. Typical records that show an invalid password for smith and guest follow:
   smith:*:100:100:8A-74(office):/home/smith:/usr/bin/sh
   guest:*:200:0::/home/guest:/usr/bin/sh

   The fields are in the following order: user name, password, user ID, primary group, general (gecos) information, home directory, and initial program (login shell). The * (asterisk) in the password field indicates that the password is invalid. Each attribute is separated by a : (colon).

2. If the password for smith in the previous example is changed to a valid password, the record will change to the following:
   smith:!100:100:8A-74(office):/home/smith:/usr/bin/sh

   The ! (exclamation point) indicates that an encrypted password is stored in the /etc/security/passwd file.

Files

/etc/passwd
Contains basic user attributes.

/usr/lib/security/mkuser.default
Contains default attributes for new users.

/etc/group
Contains the basic attributes of groups.

/etc/security/group
Contains the extended attributes of groups.

/etc/security/passwd
Contains password information.

/etc/security/user
Contains the extended attributes of users.

/etc/security/environment
Contains the environment attributes of users.

/etc/security/limits
Contains the process resource limits of users.

Related Information

The chfn command, chsh command, chuser command, lsuser mkuser command, passwd command, pwdadm command, pwdck command, rmuser command.

The endpwent subroutine, enduserdb subroutine, getpwent subroutine, getpwnam subroutine, getpwuid subroutine, getuserattr subroutine, IDtouser subroutine, nextuser subroutine, putpwent subroutine, putuserattr subroutine, setuserdb subroutine.


passwd_policy File

Purpose

Defines the types and manifest constants required to support the passwdpolicy() function.

Description

The passwdpolicy() interface uses named policies to determine the strength of a new password. This interface is intended for applications which maintain policy information in either the per-user attribute databases (for example /etc/security/user) or which use the new policy files (/etc/security/passwd_policy and /usr/lib/security/passwd_policy) to create password policies which are not associated with a specific user.

System security applications may name policies which are then enforced using the named rules in /etc/security/passwd_policy. Because this file resides in the /etc/security directory, it is only accessible by applications run by root or a member of group security. The /usr/lib/security/passwd_policy file is intended for applications which wish to use these new APIs to enforce their own password strength rules.
There is no support for a default stanza, rather each application must name a stanza it wishes to use as the default and then explicitly check against that stanza. In addition to the basic construction rules which are supported by `passwdstrength()`, this interface supports dictionary checking, per-user password histories, and administrator-defined load module extensions.

The format of the `passwd_policy` file is similar to the password construction rule attributes as stored in the `/etc/security/user` file, with the exception that named policies do not include the histsize and histexpire attributes. Each file is a sequence of zero or more stanzas with the named policy being the stanza name. Each stanza contains one or more attributes describing the password rules which must be satisfied for a password to be accepted.

Example

```
ibm_corp_policy:
  dictionlist = /usr/share/dict/words,/usr/local/lib/local_words
  maxage = 26
  minage = 2
  maxexpired = 2
  maxrepeats = 2
  mindiff = 6
  minalpha = 4
  minother = 2
  minlen = 7
  pwdchecks = /usr/lib/security/more_checks.so
```

The maxage, minage, maxexpired, maxrepeats, mindiff, minalpha, minother, and minlen attributes are integers. The dictionlist and pwdchecks attributes are comma-separated lists of filenames. For more information on valid values for attributes, please see `/etc/security/user`.

Permissions

The permissions on `/etc/security/passwd_policy` shall be 660, owner root, group security. This restricts access to processes with the privileges needed to perform other security administrative tasks. The permissions on `/usr/lib/security/passwd_policy` shall be 664, owner root, group security. This allows all processes to read the file, while restricting administrative access to processes with the privileges needed to perform other security administrative tasks. Applications select between policy files using the type parameter to the `passwdpolicy()` function.

Location

```
/usr/lib/security/passwd_policy Location of policy values for PWP_LOCALPOLICY.
/etc/security/passwd_policy Location of policy values for PWP_SYSTEMPOLICY.
```

Related Information

The `/etc/security/user` file.

The `/usr/include/pwdpolicy.h` file.

/`etc/security/passwd` File

Purpose

Contains password information.
Description

The `/etc/security/passwd` file is an ASCII file that contains stanzas with password information. Each stanza is identified by a user name followed by a `:` (colon) and contains attributes in the form `Attribute=Value`. Each attribute is ended with a new line character, and each stanza is ended with an additional new line character.

Each stanza can have the following attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>password</td>
<td>Specifies the encrypted password. The system encrypts the password created with the <code>passwd</code> command or the <code>pwdadm</code> command. If the password is empty, the user does not have a password. If the password is an <code>*</code> (asterisk), the user cannot log in. The value is a character string. The default value is <code>*</code>.</td>
</tr>
<tr>
<td>lastupdate</td>
<td>Specifies the time (in seconds) since the epoch (00:00:00 GMT, January 1, 1970) when the password was last changed. If password aging (the <code>minage</code> attribute or the <code>maxage</code> attribute) is in effect, the <code>lastupdate</code> attribute forces a password change when the time limit expires. (See the <code>/etc/security/user</code> file for information on password aging.) The <code>passwd</code> and <code>pwdadm</code> commands normally set this attribute when a password is changed. The value is a decimal integer that can be converted to a text string using the <code>ctime</code> subroutine.</td>
</tr>
<tr>
<td>flags</td>
<td>Specifies the restrictions applied by the <code>login</code>, <code>passwd</code>, and <code>su</code> commands. The value is a list of comma-separated attributes. The <code>flags</code> attribute can be left blank or can be one or more of the following values:</td>
</tr>
<tr>
<td></td>
<td><strong>ADMIN</strong> Defines the administrative status of the password information. If the <code>ADMIN</code> attribute is set, only the root user can change this password information.</td>
</tr>
<tr>
<td></td>
<td><strong>ADMCHG</strong> Indicates that the password was last changed by a member of the security group or the root user. Normally this flag is set implicitly when the <code>pwdadm</code> command changes another user's password. When this flag is set explicitly, it forces the password to be updated the next time a user gives the <code>login</code> command or the <code>su</code> command.</td>
</tr>
<tr>
<td></td>
<td><strong>NOCHECK</strong> None of the system password restrictions defined in the <code>/etc/security/user</code> file are enforced for this password.</td>
</tr>
</tbody>
</table>

When the `passwd` or `pwdadm` command updates a password, the command adds values for the `password` and `lastupdate` attributes and, if used to change another user’s password, for the `flags ADMCHG` attribute.

Access to this file should be through the system commands and subroutines defined for this purpose. Other accesses may not be supported in future releases. Users can update their own passwords with the `passwd` command, administrators can set passwords and password flags with the `pwdadm` command, and the root user is able to use the `passwd` command to set the passwords of other users.

Refer to the “Files” section for information on where attributes and other information on users and groups are stored.

Although each user name must be in the `/etc/passwd` file, it is not necessary to have each user name listed in the `/etc/security/passwd` file. If the authentication attributes `auth1` and `auth2` are so defined in the `/etc/security/user` file, a user may use the authentication name of another user. For example, the authentication attributes for user tom can allow that user to use the entry in the `/etc/security/passwd` file for user carol for authentication.

Security

Access Control: This file should grant read (r) and write (w) access only to the root user.
Auditing Events:

<table>
<thead>
<tr>
<th>Event</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>S_PASSWD_READ</td>
<td>file name</td>
</tr>
<tr>
<td>S_PASSWD_WRITE</td>
<td>file name</td>
</tr>
</tbody>
</table>

**Examples**

The following line indicates that the password information in the `/etc/security/passwd` file is available only to the root user, who has no restrictions on updating a password for the specified user:

```plaintext
flags = ADMIN,NOCHECK
```

An example of this line in a typical stanza for user smith follows:

```plaintext
smith:
  password = MGURSj.F056Dj
  lastupdate = 62307865
  flags = ADMIN,NOCHECK
```

The password line shows an encrypted password. The lastupdate line shows the number of seconds since the epoch that the password was last changed. The flags line shows two flags: the **ADMIN** flag indicates that the information is available only to the root user, and the **NOCHECK** flag indicates that the root user has no restrictions on updating a password for the specified user.

**Files**

- `/etc/security/passwd` Specifies the path to the file.
- `/etc/passwd` Contains basic user attributes.
- `/etc/security/user` Contains the extended attributes of users.
- `/etc/security/login.cfg` Contains configuration information for login and user authentication.

**Related Information**

The **login** command, **passwd** command, **pwdadm** command, **su** command.

The **ftpd** daemon, **rlogind** daemon.

The **ctime** subroutine, **endpwdb** subroutine, **getuserpw** subroutine, **putuserpw** subroutine, **setpwdb** subroutine.

**pcnfsd.conf Configuration File**

**Purpose**

Provides configuration options for the **rpc.pcnfsd** daemon.

**Description**

The `/etc/pcnfsd.conf` file is an ASCII file written by users to add options to the operation of the **rpc.pcnfsd** daemon, which takes no command-line flags. This file is part of Network Support Facilities.

When started, the **rpc.pcnfsd** daemon checks for the presence of the **pcnfsd.conf** configuration file and conforms its performance to the specified arguments. The following options can be entered in the **pcnfsd.conf** file:
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aixargs -B CharacterPair</td>
<td>Controls the printing of burst pages according to the value of the_CharacterPair_variable, as listed below. The first character applies to the header and the second character to the trailer. Possible values are n (never), a (always), and g (group).</td>
</tr>
<tr>
<td>HT</td>
<td>Description</td>
</tr>
<tr>
<td>nn</td>
<td>No headers, no trailers</td>
</tr>
<tr>
<td>na</td>
<td>No headers, trailer on every file</td>
</tr>
<tr>
<td>ng</td>
<td>No header, trailer at the end of the job</td>
</tr>
<tr>
<td>an</td>
<td>Header on every file, no trailers</td>
</tr>
<tr>
<td>aa</td>
<td>Headers and trailers on every file in the job</td>
</tr>
<tr>
<td>ag</td>
<td>Header on every file, trailer after job</td>
</tr>
<tr>
<td>gn</td>
<td>Header at beginning of job, no trailer</td>
</tr>
<tr>
<td>ga</td>
<td>Header at beginning of job, trailer after every file</td>
</tr>
<tr>
<td>gg</td>
<td>Header at beginning of job, trailer at end of job</td>
</tr>
</tbody>
</table>

The header and trailer stanzas in the `/etc/qconfig` file define the default treatment of burst pages.

**Note:** The `-B` flag works exactly like the `-B` flag in the `enq` command. Unlike the `enq` command, however, the `rpc.pcnfsd` daemon does not allow spaces between the `-B` flag and the _CharacterPair_ variable.

Disables the `rpc.pcnfsd` daemon feature that returns job numbers when print jobs are submitted.

getjobnum off
Option

printer Name AliasFor Command

Description

Defines a PC-NFS virtual printer, recognized only by rpc.pcnfsd daemon clients. Each virtual printer is defined on a separate line in the pcnfsd.conf file. The following variables are specified with this option.

Name

Specifies the name of the PC-NFS virtual printer to be defined.

AliasFor

Specifies the name of an existing printer that performs the print job.

Note: To define a PC-NFS virtual printer associated with no existing printer, use a single - (minus sign) instead of the AliasFor variable.

Command

Specifies the command that is run when a file is printed on the Name printer. This command is executed by the Bourne shell, using the -c option. For complex operations, replace the Command variable with an executable shell script.

The following list of tokens and substitution values can be used in the Command variable:

Token Substitution Value

$FILE The full path name of the print data file. After the command has executed, the file will be unlinked.

$USER The user name of the user logged in to the client.

$HOST The host name of the client system.

spooldir PathName

Designates a new parent directory, PathName, where the rpc.pcnfsd daemon stores the subdirectories it creates for each of its clients. The default parent directory is /var/spool/pcnfs.

uidrange

Specifies the valid UID (user number) range that the rpc.pcnfsd daemon accepts. The default UID range is 101-4294967295.

wtmp off

Disables the login record-keeping feature of the rpc.pcnfsd daemon. By default, the daemon appends to the /var/adm/wtmp file a record of user logins.

Examples

1. The following sample pcnfsd.conf configuration file demonstrates the effects some options have on the operation of the rpc.pcnfsd daemon:

   printer test - /usr/bin/cp $FILE
   /usr/tmp/$HOST-$USER
   printer sandman san ls -l $FILE
   wtmp off

   The first line establishes a printer test. Files sent to the test printer will be copied into the /usr/tmp directory. Requests to the test PC-NFS virtual printer to list the queue, check the status, or do similar printer operations, will be rejected because a - (minus sign) has been given for the Alias-For parameter.
The second line establishes a PC-NFS virtual printer called sandman that lists, in long form, the file specifications for the print data file.

The third line turns off the rpc.pcnfsd daemon feature that records user logins.

2. To set a UID range enter:

```
uidrange 1-100,200-50000
```

This entry means that only numbers from 101-199 and over 50000 are invalid UID numbers.

**Files**

- `/etc/pcnfsd.conf` Specifies the path of the configuration file.
- `/var/spool/pcnfs` directory Contains subdirectories for clients of the pcnfsd daemon.
- `/etc/qconfig` Configures a printer queuing system.
- `/var/adm/wtmp` Describes formats for user and accounting information.

**Related Information**

The `enq` command.

The `rpc.pcnfsd` daemon.

Bourne shell in the *Operating system and device management*.

Network File System Overview and List of NFS files in the *Networks and communication management*.

**pkginfo File**

**Purpose**

Describes the format of a package characteristics file.

**Description**

The `pkginfo` file is an ASCII file that describes the characteristics of the package along with information that helps control the flow of installation. It is created by the software package developer.

Each entry in the `pkginfo` file is a line that establishes the value of a parameter in the following form:

```
PARAM="value"
```

There is no required order in which the parameters must be specified within the file. Each parameter is described below. Only fields marked with an asterisk are mandatory.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PKG*</td>
<td>PKG is the parameter to which you assign an abbreviation for the name of the package being installed. The abbreviation must be a short string (no more than nine characters long) and it must conform to file naming rules. All characters in the abbreviation must be alphanumeric and the first cannot be numeric. <code>install</code>, <code>new</code>, and <code>all</code> are reserved abbreviations.</td>
</tr>
<tr>
<td>NAME*</td>
<td>Text that specifies the package name (maximum length of 256 ASCII characters).</td>
</tr>
<tr>
<td>ARCH*</td>
<td>A comma-separated list of alphanumeric tokens that indicate the architecture (for example, <code>ARCH=m68k,i386</code>) associated with the package. The <code>pkgmk(1M)</code> tool can be used to create or modify this value when actually building the package. The maximum length of a token is 16 characters and it cannot include a comma.</td>
</tr>
<tr>
<td>VERSION*</td>
<td>Text that specifies the current version associated with the software package. The maximum length is 256 ASCII characters and the first character cannot be a left parenthesis. The <code>pkgmk</code> tool can be used to create or modify this value when actually building the package.</td>
</tr>
<tr>
<td><strong>Parameter</strong></td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>--------------</td>
<td>----------------</td>
</tr>
<tr>
<td>CATEGORY*</td>
<td>A comma-separated list of categories under which a package can be displayed. There are six categories: &quot;application,&quot; &quot;graphics,&quot; &quot;system,&quot; &quot;utilities,&quot; &quot;set,&quot; and &quot;patch.&quot; If you choose, you can also assign a package to one or more categories that you define. Categories are case-insensitive and can contain only alphanumerics. Each category is limited in length to 16 characters. For a Set Installation Package (SIP), this field must have the value &quot;set.&quot; A SIP is a special purpose package that controls the installation of a set of packages.</td>
</tr>
<tr>
<td>DESC</td>
<td>Text that describes the package (maximum length of 256 ASCII characters).</td>
</tr>
<tr>
<td>VENDOR</td>
<td>Used to identify the vendor that holds the software copyright (maximum length of 256 ASCII characters).</td>
</tr>
<tr>
<td>HOTLINE</td>
<td>Phone number and/or mailing address where further information can be received or bugs can be reported (maximum length of 256 ASCII characters).</td>
</tr>
<tr>
<td>EMAIL</td>
<td>An electronic address where further information is available or bugs can be reported (maximum length of 256 ASCII characters).</td>
</tr>
<tr>
<td>VSTOCK</td>
<td>The vendor stock number, if any, that identifies this product (maximum length of 256 ASCII characters).</td>
</tr>
<tr>
<td>CLASSES</td>
<td>A space-separated list of classes defined for a package. The order of the list determines the order in which the classes are installed. Classes listed first are installed first (on a medium-by-medium basis). This parameter can be modified by the request script. In this way, the request script can be used to select which classes in the package get installed on the system.</td>
</tr>
<tr>
<td>ISTATES</td>
<td>A list of allowable run states for package installation (for example, &quot;'S s 1''&quot;).</td>
</tr>
<tr>
<td>RSTATES</td>
<td>A list of allowable run states for package removal (for example, &quot;'S s 1''&quot;).</td>
</tr>
<tr>
<td>BASEDIR</td>
<td>The pathname to a default directory where &quot;relocatable&quot; files can be installed. If BASEDIR is not specified and basedir in the admin file (/var/sadm/install/admin/default) is set to default, then BASEDIR is set to / by default. An administrator can override the value of BASEDIR by setting basedir in the admin file.</td>
</tr>
<tr>
<td>ULIMIT</td>
<td>If set, this parameter is passed as an argument to the ulimit command, which establishes the maximum size of a file during installation.</td>
</tr>
<tr>
<td>ORDER</td>
<td>A list of classes defining the order in which they should be put on the medium. Used by pkgmk(1) in creating the package. Classes not defined in this field are placed on the medium using the standard ordering procedures.</td>
</tr>
<tr>
<td>PSTAMP</td>
<td>Production stamp used to mark the pkgmap(4) file on the output volumes. Provides a means for distinguishing between production copies of a version if more than one is in use at a time. If PSTAMP is not defined, the default is used. The default consists of the UNIX system machine name followed by the string &quot;'YYMMDDHHmm'&quot; (year, month, date, hour, minutes).</td>
</tr>
<tr>
<td>INTONLY</td>
<td>Indicates that the package should be installed interactively only when set to any non-NULL value.</td>
</tr>
<tr>
<td>PREDEPEND</td>
<td>Used to maintain compatibility with dependency checking on packages delivered earlier than System V Release 4. Pre-Release 4 dependency checks were based on whether or not the name file for the required package existed in the /usr/options directory. This directory is not maintained for Release 4 and later packages because the depend file is used for checking dependencies. However, entries can be created in this directory to maintain compatibility. This is done automatically by pkgmk. This field is to be assigned the package instance name of the package.</td>
</tr>
<tr>
<td>SERIALNUM</td>
<td>A serial number, if any, that uniquely identifies this copy of the package (maximum length of 256 ASCII characters).</td>
</tr>
<tr>
<td>ACTKEY</td>
<td>Activation key indicator. Set to YES indicates that an activation key is required to install the package.</td>
</tr>
<tr>
<td>PRODUCTNAME</td>
<td>A list of the products to which each package belongs. The format of this variable is: PRODUCTNAME=&quot;&lt;product&gt;[[&lt;product&gt;] . . . ]&quot;</td>
</tr>
</tbody>
</table>

Developers can define their own installation parameters by adding a definition to this file. A developer-defined parameter should begin with a capital letter.
Restrictions placed on a package installation by certain variables in the pkginfo file can be overridden by instructions in the admin file. For example, the restriction of allowable run states set by the ISTATES variable can be overridden by having

```
runlevel=nocheck
```

in the admin file being used for installation. (Default is “default”.) See the admin file for further information.

**Examples**

Here is a sample pkginfo file:

```
PKG="oam"
NAME="OAM Installation Utilities"
VERSION="3"
VENDOR="AT&T"
HOTLINE="1-800-ATT-BUGS"
EMAIL="attunix!olsen"
VSTOCK="0122c3f5566"
CATEGORY="system.essential"
ISTATES="S 2"
RSTATES="S 2"
```

**Related Information**

The admin file format.

---

**pkgmap File**

**Purpose**

Describes the format of a package contents description file.

**Description**

The pkgmap file is an ASCII file that provides a complete listing of the package contents. Each entry in pkgmap describes a single “deliverable object file.” A deliverable object file includes shell scripts, executable objects, data files, and directories. The entry consists of several fields of information, each field separated by a space. The fields are described below and must appear in the order shown.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>part</td>
<td>A field designating the part number in which the object resides. A part is a collection of files, and is the atomic unit by which a package is processed. A developer can choose the criteria for grouping files into a part (for example, based on class). If no value is defined in this field, part 1 is assumed.</td>
</tr>
</tbody>
</table>
Field | Description
--- | ---
ftype | A one-character field that indicates the file type. Valid values are:
  f | a standard executable or data file
  e | a file to be edited upon installation or removal
  v | volatile file (one whose contents are expected to change)
  d | directory
  x | an exclusive directory
  l | linked file
  p | named pipe
  c | character special device
  b | block special device
  i | installation script or information file
  s | symbolic link

Once a file has the file type attribute v, it will always be volatile. For example, if a file being installed already exists and has the file type attribute v, then even if the version of the file being installed is not specified as volatile, the file type attribute remains volatile.

class | The installation class to which the file belongs. This name must contain only alphanumeric characters and be no longer than 12 characters. It is not specified if the ftype is i (information file).

pathname | The pathname where the object resides on the target machine, such as /usr/bin/mail. Relative pathnames (those that do not begin with a slash) indicate that the file is relocatable.

For linked files (ftype is either l or s), pathname must be in the form of path1=path2, with path1 specifying the destination of the link and path2 specifying the source of the link.

For symbolically linked files, when path2 is a relative pathname starting with ./ or ../../../, path2 is not considered relocatable. For example, if you enter a line such as

```
  s /foo/bar/etc/mount=../usr/sbin/mount
```

path1 (/foo/bar/etc/mount) is a symbolic link to ../../../usr/sbin/mount.

pathname can contain variables which support relocation of the file. A "$" parameter can be embedded in the pathname structure. SBASEDIR can be used to identify the parent directories of the path hierarchy, making the entire package easily relocatable. Default values for parameter and BASEDIR must be supplied in the pkginfo file and can be overridden at installation.

Special characters, such as an equal sign ("="), are included in pathnames by surrounding the entire pathname in single quotes (as in, for example, '/usr/lib/~=').

major | The major device number. The field is only specified for block or character special devices.

minor | The minor device number. The field is only specified for block or character special devices.

mode | The octal mode of the file (for example, 0664). A question mark ("?"") indicates that the mode is left unchanged, implying that the file already exists on the target machine. This field is not used for linked files, packaging information files or non-installable files.

owner | The owner of the file (for example, bin or root). The field is limited to 14 characters in length. A question mark ("?"") indicates that the owner is left unchanged, implying that the file already exists on the target machine. This field is not used for linked files or non-installable files. It is used optionally with a package information file. If used, it indicates with what owner an installation script is executed.

The owner can be a variable specification in the form of $[A-Z] and is resolved at installation time.
### Field Description

**group**
The group to which the file belongs (for example, `bin` or `sys`). The field is limited to 14 characters in length. A question mark ("?"), indicates that the group is left unchanged, implying that the file already exists on the target machine. This field is not used for linked files or non-installable files. It is used optionally with a package information file. If used, it indicates with what group an installation script is executed.

Can be a variable assignment in the form of `$[A-Z]$` and is resolved at installation time.

**size**
The actual size of the file in bytes. This field is not specified for named pipes, special devices, directories, or linked files.

**cksum**
The checksum of the file contents. This field is not specified for named pipes, special devices, directories, or linked files.

**modtime**
The time of last modification. This field is not specified for named pipes, special devices, directories, or linked files.

The following three optional fields must be used as a group. That is, all three must be specified if any is specified.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>mac</strong></td>
<td>The Mandatory Access Control (MAC) Level Identifier (LID), an integer value that specifies a combination of a hierarchical classification and zero or more non-hierarchical categories. A question mark (&quot;?&quot;), indicates that the mac field is to be left unchanged, implying that the file already exists on the target machine. This field can only be applied to a file on an sfs filesystem and is not used for linked files or packaging information files. Note: Mandatory Access Control is not supported in this release; this field is present for compatibility with earlier releases only. A value of 0 should be used if you must specify this field.</td>
</tr>
<tr>
<td><strong>fixed</strong></td>
<td>A comma-separated list of valid mnemonic fixed privilege names as defined for the filepriv command. The string NULL is used in place of the comma-separated list when fixed privileges are not to be specified. A question mark (&quot;?&quot;), indicates that the fixed field is to be left unchanged, implying that the file already exists on the target machine. If the fixed attribute is not supplied, then files are installed with no fixed privileges. This field is not used for linked files or packaging information files. Note: Fixed privileges have no effect in the current release. This capability is maintained solely for compatibility with earlier releases.</td>
</tr>
<tr>
<td><strong>inherited</strong></td>
<td>A comma-separated list of valid mnemonic inherited privilege names as defined for the filepriv command. The string NULL is used in place of the comma separated list when privilege is not to be specified. A question mark (&quot;?&quot;), indicates that the inherited field is to be left unchanged, implying that the file already exists on the target machine. If the inherited attribute is not supplied, then files are installed with no inheritable privileges. This field is not used for linked files or packaging information files. Note: Inheritable privileges have no effect in the current release. This capability is maintained solely for compatibility with earlier releases.</td>
</tr>
</tbody>
</table>

Each **pkgmap** must have one line that provides information about the number and maximum size (in 512-byte blocks) of parts that make up the package. This line is in the following format:

```
:number_of_parts  maximum_part_size
```

Lines that begin with ""#"" are comment lines and are ignored.

When files are saved during installation before they are overwritten, they are normally just copied to a temporary pathname. However, for files whose mode includes execute permission (but which are not editable), the existing version is linked to a temporary pathname and the original file is removed. This allows processes which are executing during installation to be overwritten.

The **pkgmap** file can contain only one entry per unique pathname.
An exclusive directory type (*file*) type *x* specifies directories that are constrained to contain only files that appear in the installation software database (*/var/sadm/install/contents*). If there are other files in the directory, they are removed by *pkgchk -fx* as described on the manual page for the *pkgchk* command.

Variable specifications for the *owner* and *group* fields are defined in the *pkginfo* file. For example, *owner* could be *SONWER* in the *pkgmap* file; if *OWNER* is defined as *root* in the *pkginfo* file, *SONWER* gets the value *root* when the file is installed.

Examples
The following is an example *pkgmap* file.

```
:2 500
 1 i pkginfo 237 1179 541296672
 1 b class1 /dev/diskette 17 134 0644 root other
 1 c class1 /dev/rdiskette 17 134 0644 root other
 1 d none bin 0755 root bin
 1 f none bin/INSTALL 0755 root bin 11103 17954 541295535
 1 f none bin/REMOVE 0755 root bin 3214 50237 541295541
 1 l none bin/UNINSTALL=bin/REMOVE
 1 f none bin/cmda 0755 root bin 3580 60325 541295567 0 NULL
 1 f none bin/cmdb 0755 root bin 49107 51255 541438368
 1 f class1 bin/cmdb 0755 root bin 45599 26048 541295599
 1 f class1 bin/cmd 0755 root bin 4648 8473 541461238
 1 f none bin/cmde 0755 root bin 40501 1264 541295622
 1 f class2 bin/cmdf 0755 root bin 2345 35889 541295574
 1 f none bin/cmdg 0755 root bin 41185 47653 541461242
 2 d class2 data 0755 root bin
 2 p class1 data/apipe 0755 root other
 2 d none log 0755 root bin 0 NULL NULL
 2 v none log/logfile 0755 root bin 41815 47563 541461333
 2 d none save 0755 root bin
 2 d none spool 0755 root bin
 2 d none tmp 0755 root bin
```

Related Information
The [pkgchk](#) command.

---

**policy.cfg File**

**Purpose**
The *policy.cfg* file contains attributes that are used while creating certificates when creating users or adding certificates to the local LDAP repository.

**Description**
The *policy.cfg* file consists of four stanzas: *newuser*, *storage*, *crl* and *comm*. These stanzas modify the behavior of some system administration commands. The *mkuser* command uses the *newuser* stanza. The *certlink* command uses the *storage* stanza. The *certadd* and *certlink* command use the *comm* and *crl* stanzas.

**Examples**

```
*******************************************************************************
* Example policy.cfg file
*******************************************************************************
* newuser Stanza:
*
* cert Specifies whether the mkuser command generates a certificate (new) or
*     not (get) by default.
```
* ca Specifies the CA used by the mkuser command when generating
  a certificate.
* version Specifies the version number of the certificate to be created.
  The value 3 is the only supported value.
* tag Specifies the auth_cert tag value used by the mkuser command when
  creating a user when cert = new.
* label Specifies the private key label used by the mkuser command when
  generating a certificate.
* keystore Specifies the keystore URI used by the mkuser command when generating
  a certificate.
* passwd Specifies the keystore's password used by the mkuser command when
  generating a certificate.
* domain Specifies the domain part of the certificate's subject alternate name
  email value used by the mkuser command when generating a
  certificate.
* validity Specifies the certificate's validity period value used by the mkuser
  command when generating a certificate.
* algorithm Specifies the public key algorithm used by the mkuser command when
  generating a certificate.
* keysize Specifies the minimum encryption key size in bits used by the mkuser
  command when generating a certificate.
* keyusage Specifies the certificate's key usage value used by the mkuser
  command when generating a certificate.
* subalturi Specifies the certificate's subject alternate name URI value
  used by the mkuser command when generating a certificate.

* storage Stanza:

  * replicate Specifies whether the certlink command saves a copy of the certificate
    (yes) or just the link (no).

* crl Stanza

  * check Specifies whether the certadd and certlink commands should check the
    CRL (yes) or not (no).

* comm Stanza

  * timeout Specifies the timeout period in seconds when requesting certificate
    information using HTTP (e.g., retrieving CRLs).

newuser:
  cert = new
  ca = local
  passwd = pki
  version = "3"
  keysize = 1024
  keystore = test
  validity = 60

storage:
  replicate = no

crl:
  check = yes

comm:
  timeout = 10

* end of policy.cfg

File
/usr/lib/security/pki/policy.cfg
Related Information

The `mkuser`, `certcreate`, `certrevoke`, `certadd`, and `certlink` commands.

The `/usr/lib/security/pki/acct.cfg` and `/usr/lib/security/pki/ca.cfg` files.

---

**portlog File**

**Purpose**
Contains per-port unsuccessful login attempt information and port locks.

**Description**
The `/etc/security/portlog` file is an ASCII file that contains stanzas of per port unsuccessful login attempt information and port locks. Each stanza has a name followed by a : (colon) that defines the port name. Attributes are in the form `Attribute=Value`. Each attribute ends with a new line character and each stanza ends with an additional new line character.

The attributes in the stanzas are as follows:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>locktime</td>
<td>Defines the time the port was locked in seconds since the epoch (zero time, January 1, 1970). This value is a decimal integer string.</td>
</tr>
<tr>
<td>unsuccessful_login_times</td>
<td>Lists the times of unsuccessful login attempts in seconds since the epoch. The list contains decimal integer strings separated by commas.</td>
</tr>
</tbody>
</table>

These attributes do not have default values. If a value is not specified, the attribute is ignored.

**Security**
Access Control: This file grants read access to the root user and members of the security group, and write access only to the root user. Access for other users and groups depends upon the security policy of the operating system.

**Examples**
A typical record looks like the following example for the `/dev/tty0` port:

```
/dev/tty0:
   locktime = 723848478
   unsuccessful_login_times = 723848430,723848450,723848478
```

**Files**

`/etc/security/portlog` Specifies the path to the file.
`/etc/security/login.cfg` Contains configuration information for login and user authentication.

**Related Information**
The `chsec` command, `login` command, `su` command.

The `loginfailed` subroutine, `loginrestrictions` subroutine.

Security Administration in *Operating system and device management.*
/proc File

Purpose
Contains state information about processes and threads in the system.

Syntax
```
#include <sys/procfs.h>
```

Description
The /proc file system provides access to the state of each active process and thread in the system. The name of each entry in the /proc file system is a decimal number corresponding to the process ID. These entries are subdirectories and the owner of each is determined by the user ID of the process. Access to the process state is provided by additional files contained within each subdirectory. Except where otherwise specified, the term /proc file is meant to refer to a non-directory file within the hierarchy rooted at /proc. The owner of each file is determined by the user ID of the process.

The various /proc directory, file, and field names contain the term lwp (light weight process). This term refers to a kernel thread. The /proc files do not refer to user space pthreads. While the operating system does not use the term lwp to describe its threads, it is used in the /proc file system for compatibility with other UNIX operating systems.

The following standard subroutine interfaces are used to access the /proc files:
- open subroutine
- close subroutine
- read subroutine
- write subroutine

Most files describe process state and are intended to be read-only. The ctl (control) and lwpctl (thread control) files permit manipulation of process state and can only be opened for writing. The as (address space) file contains the image of the running process and can be opened for both reading and writing. A write open allows process control while a read-only open allows inspection but not process control. Thus, a process is described as open for reading or writing if any of its associated /proc files is opened for reading or writing, respectively.

In general, more than one process can open the same /proc file at the same time. Exclusive open is intended to allow process control without another process attempting to open the file at the same time. A process can obtain exclusive control of a target process if it successfully opens any /proc file in the target process for writing (the as or ctl files, or the lwpctl file of any kernel thread) while specifying the O_EXCL flag in the open subroutine. Such a call of the open subroutine fails if the target process is already open for writing (that is, if a ctl, as, or lwpctl file is open for writing). Multiple concurrent read-only instances of the open subroutine can exist; the O_EXCL flag is ignored on the open subroutine for reading. The first open for writing by a controlling process should use the O_EXCL flag. Multiple processes trying to control the same target process usually results in errors.

Data may be transferred from or to any locations in the address space of the traced process by calling the lseek subroutine to position the as file at the virtual address of interest, followed by a call to the read or write subroutine. An I/O request extending into an unmapped area is truncated at the boundary. A read or write request beginning at an unmapped virtual address fails with errno set to EFAULT.

Information and control operations are provided through additional files. The <sys/procfs.h> file contains definitions of data structures and message formats used with these files. Some of these definitions use sets of flags. The set types pr_sigset_t, fltset_t, and syset_t correspond to signal, fault, and system call enumerations, respectively. These enumerations are defined in the <sys/procfs.h> files. The pr_sigset_t
and fltset_t types are large enough to hold flags for its own enumeration. Although they are of different sizes, they have a common structure and can be manipulated by the following macros:

```
prfillset(&set); /* turn on all flags in set */
premptyset(&set); /* turn off all flags in set */
praddset(&set, flag); /* turn on the specified flag */
prdelset(&set, flag); /* turn off the specified flag */
r = prismember(&set, flag); /* != 0 if flag is turned on */
```

Either the prfillset or premptyset macro must be used to initialize the pr_sigset_t or fltset_t type before it is used in any other operation. The flag parameter must be a member of the enumeration that corresponds to the appropriate set.

The sysset_t set type has a different format, set of macros, and a variable length structure to accommodate the varying number of available system calls. You can determine the total number of system calls, their names, and number of each individual call by reading the sysent file. You can then allocate memory for the appropriately sized sysset_t structure, initialize its pr_size field, and then use the following macros to manipulate the system call set:

```
prfillsysset(&set) /* set all syscalls in the sysset */
premptysysset(&set) /* clear all syscalls in the sysset */
praddsysset(&set, num) /* set specified syscall in the sysset */
predelsysset(&set, num) /* clear specified syscall in the sysset */
prissyssetmember(&set, num) /* != 0 if specified syscall is set */
```

See the description of the sysent file for more information about system calls.

Every active process contains at least one kernel thread. Every kernel thread represents a flow of execution that is independently scheduled by the operating system. All kernel threads in a process share address space as well as many other attributes. Using the ctl and lwpctl files, you can manipulate individual kernel threads in a process or manipulate all of them at once, depending on the operation.

When a process has more than one kernel thread, a representative thread is chosen by the system for certain process status file and control operations. The representative thread is stopped only if all the process’s threads are stopped. The representative thread may be stopped on an event of interest only if all threads are stopped, or it may be stopped by a PR_REQUESTED stop only if no other events of interest exist.

The representative thread remains fixed as long as all the threads are stopped on events of interest or are in PR_SUSPENDED stop and the PCRUN operand is not applied to any of them.

When applied to the process control file (ctl), every /proc control operation that affects a kernel thread uses the same algorithm to choose which kernel thread to act on. With synchronous stopping (see PCSET), this behavior enables an application to control a multiple thread process using only the process level status and control files. For more control, use the thread-specific lwpctl files.

The /proc file system can be used by both 32-bit and 64-bit control processes to get information about both 32-bit and 64-bit target processes. The /proc files provide 64-bit enabled mode invariant files to all observers. The mode of the controlling process does not affect the format of the /proc data. Data is returned in the same format to both 32-bit and 64-bit control processes. Addresses and applicable length and offset fields in the /proc files are 8 bytes long.

**Directory Structure**

At the top level, the /proc directory contains entries, each of which names an existing process in the system. The names of entries in this directory are process ID (pid) numbers. These entries are directories. Except where otherwise noted, the files described below are read-only. In addition, if a process becomes a zombie (one that has been terminated by its parent with an exit call but has not been suspended by a
wait call), most of its associated /proc files disappear from the directory structure. Normally, later attempts to open or to read or write to files that are opened before the process is terminated elicit the ENOENT message. Exceptions are noted.

The /proc files contain data that presents the state of processes and threads in the system. This state is constantly changing while the system is operating. To lessen the load on system performance caused by reading /proc files, the /proc file system does not stop system activity while gathering the data for those files. A single read of a /proc file generally returns a coherent and fairly accurate representation of process or thread state. However, because the state changes as the process or thread runs, multiple reads of /proc files may return representations that show different data and therefore appear to be inconsistent with each other.

An atomic representation is a representation of the process or thread at a single and discrete point in time. If you want an atomic snapshot of process or thread state, stop the process and thread before reading the state. There is no guarantee that the data is an atomic snapshot for successive reads of /proc files for a running process. In addition, a representation is not guaranteed to be atomic for any I/O applied to the as (address space) file. The contents of any process address space might be simultaneously modified by a thread of that process or any other process in the system.

**Note:** Multiple structure definitions are used to describe the /proc files. A /proc file may contain additional information other than the definitions presented here. In future releases of the operating system, these structures may grow by the addition of fields at the end of the structures.

**The /proc/pid File Structure**
The /proc/pid directory contains (but is not limited to) the following entries:

- **as** Contains the address space image of the process. The as file can be opened for both reading and writing. The lseek subroutine is used to position the file at the virtual address of interest. Afterwards, you can view and modify the address space with the read and write subroutines, respectively.

- **ctl** A write-only file to which structured messages are written directing the system to change some aspect of the process’s state or control its behavior in some way. The seek offset is not relevant when writing to this file, see types of control messages for more information. Individual threads also have associated lwpctl files. A control message may be written either to the ctl file of the process or to a specific lwpctl file with operation-specific effects as described. The effect of a control message is immediately reflected in the state of the process visible through appropriate status and information files.

- **status** Contains state information about the process and one of its representative thread. The file is formatted as a struct pstatus type containing the following members:

```c
uint32_t pr_flag; /* process flags from proc struct p_flag */
uint32_t pr_flag2; /* process flags from proc struct p_flag2 */
uint32_t pr_flags; /* /proc flags */
uint32_t pr_nlwp; /* number of threads in the process */
char pr_stat; /* process state from proc p_stat */
char pr_dmodel; /* data model for the process */
char pr__pad1[6]; /* reserved for future use */
prSigset_t pr_psigpend; /* set of process pending signals */
prptr64_t pr_brkbase; /* address of the process heap */
uint64_t pr_brksize; /* size of the process heap, in bytes */
prptr64_t pr_stkbase; /* address of the process stack */
uint64_t pr_stksize; /* size of the process stack, in bytes */
pid64_t pr_pid; /* process id */
pid64_t pr_ppid; /* parent process id */
pid64_t pr_pgid; /* process group id */
pid64_t pr_sid; /* session id */
```
struct pr_timestruc64_t pr_utime; /* process user cpu time */
struct pr_timestruc64_t pr_stime; /* process system cpu time */
struct pr_timestruc64_t pr_cutime; /* sum of children's user times */
struct pr_timestruc64_t pr_cstime; /* sum of children's system times */
pr_sigset_t pr_sigtrace; /* mask of traced signals */
fltset_t pr_flttrace; /* mask of traced hardware faults */
uint32_t pr_sysentry_offset; /* offset into pstatus file of sysset_t
               * identifying system calls traced on
               * entry. If 0, then no entry syscalls
               * are being traced. */
uint32_t pr_sysexit_offset; /* offset into pstatus file of sysset_t
               * identifying system calls traced on
               * exit. If 0, then no exit syscalls
               * are being traced. */
uint64_t pr__pad[8]; /* reserved for future use */
lwpstatus_t pr_lwp; /* "representative" thread status */

The members of the status file are described below:

pr_flags
   Specifies a bit-mask holding these flags:

   **PR_ISSYS**
   Process is a kernel process (see PCSTOP)

   **PR_FORK**
   Process has its inherit-on-fork flag set (see PCSET)

   **PR_RLC**
   Process has its run-on-last-close flag set (see PCSET)

   **PR_KLC**
   Process has its kill-on-last-close flag set (see PCSET)

   **PR_ASYNC**
   Process has its asynchronous-stop flag set (see PCSET)

   **PR_PTRACE**
   Process is controlled by the ptrace subroutine

pr_nlwp
   Specifies the total number of threads in the process

pr_brkbase
   Specifies the virtual address of the process heap

pr_brksize
   Specifies the size, in bytes, of the process heap

   **Note:** The address formed by the sum of the pr_brkbase and pr_brksize is the process
   break (see the brk subroutine).

pr_stkbase
   Specifies the virtual address of the process stack

pr_stksize
   Specifies the size, in bytes, of the process stack

   **Note:** Each thread runs on a separate stack. The operating system grows the process
   stack as necessary.

pr_pid
   Specifies the process ID
pr_ppid
Specifies the parent process ID

pr_pgid
Specifies the process group ID

pr_sid
Specifies the session ID of the process

pr_utime
Specifies the user CPU time consumed by the process

pr_stime
Specifies the system CPU process time consumed by the process

pr_cutime
Specifies the cumulative user CPU time consumed by the children of the process, expressed in seconds and nanoseconds

pr_cstime
Specifies the cumulative system CPU time, in seconds and nanoseconds, consumed by the process's children

pr_sigtrace
Specifies the set of signals that are being traced (see the PCSTRACE signal)

pr_flttrace
Specifies the set of hardware faults that are being traced (see the PCSFAULT signal)

pr_sysentry_offset
If non-zero, contains offsets into the status file to the sysset_t sets of system calls being traced on system call entry (see the PCSENTRY signal). This flag is zero if system call tracing is not active for the process.

pr_sysexit_offset
If non-zero, contains offsets into the status file to the sysset_t sets of system calls being traced on system call exit (see the PCSEXIT signal). This field is zero if system call tracing is not active for the process.

pr_lwp
If the process is not a zombie, contains an lwpstatus_t structure describing a representative thread. The contents of this structure have the same meaning as if it was read from a lwpstatus file.

psinfo
Contains information about the process needed by the ps command. If the process contains more than one thread, a representative thread is used to derive the lwpinfo information. The file is formatted as a struct psinfo type and contains the following members:

uint32_t pr_flag; /* process flags from proc struct p_flag */
uint32_t pr_flag2; /* process flags from proc struct p_flag2 */
uint32_t pr_nlwp; /* number of threads in process */
uid_t pr_uid; /* real user id */
uid_t pr_euid; /* effective user id */
gid_t pr_gid; /* real group id */
gid_t pr_egid; /* effective group id */
uint32_t pr_argc; /* initial argument count */
pid64_t pr_pid; /* unique process id */
pid64_t pr_ppid; /* process id of parent */
pid64_t pr_ppid; /* pid of process group leader */
pid64_t pr_sid; /* session id */
dev64_t pr/ttydev; /* controlling tty device */
prptr64_t pr_addr; /* internal address of proc struct */
uint64_t pr_size; /* size of process image in KB (1024) units */
uint64_t pr_rssize; /* resident set size in KB (1024) units */
struct pr_timestruc64_t pr_start; /* process start time, time since epoch */
struct pr_timestruc64_t pr_time; /* usr+sys cpu time for this process */
prptr64_t pr_argv; /* address of initial argument vector in user process */
prptr64_t pr_envp; /* address of initial environment vector in user process */
char pr_fname[PRFNSZ]; /* last component of exec()ed pathname*/
char pr_psargs[PRARGSZ]; /* initial characters of arg list */
uint64_t pr__pad[8]; /* reserved for future use */
struct lwpsinfo pr_lwp; /* "representative" thread info */

Note: Some of the entries in the psinfo file, such as pr_flag, pr_flag2, and pr_addr, refer to internal kernel data structures and might not retain their meanings across different versions of the operating system. They mean nothing to a program and are only useful for manual interpretation by a user aware of the implementation details.

The psinfo file is accessible after a process becomes a zombie.

The pr_lwp flag describes the representative thread chosen. If the process is a zombie, the pr_nlwp and pr_lwp.pr_lwpid flags are zero and the other fields of pr_lwp are undefined.

map Contains information about the virtual address map of the process. The file contains an array of prmap structures, each of which describes a contiguous virtual address region in the address space of the traced process.

Note: In AIX 5.1, there may be a limited number of array entries in this file. The map file may only contain entries for virtual address regions in the process that contain objects loaded into the process. This file may not contain array entries for other regions in the process such as the stack, heap, or shared memory segments.

The prmap structure contains the following members:

uint64_t pr_size; /* size of mapping in bytes */
prptr64_t pr_vaddr; /* virtual address base */
char pr_mapname[PRMAPSZ]; /* name in /proc/pid/object */
uint64_t pr_off; /* offset into mapped object, if any */
uint32_t pr_mflags; /* protection and attribute flags */
uint32_t pr_pathoff; /* if map entry is for a loaded object, */
/* offset into the map file to a */
/* null-terminated path name followed */
/* by a null-terminated member name. */
/* If file is not an archive file, the */
/* member name is null. */
/* The offset is 0 if map entry is */
/* not for a loaded object. */

The members are described below:

pr_vaddr
  Specifies the virtual address of the mapping within the traced process

pr_size
  Specifies the size of the mapping within the traced process

pr_mapname
  If not an empty string, contains the name of a file that resides in the object directory and contains a file descriptor for the object to which the virtual address is mapped. The file is opened with the open subroutine.

pr_off
  Contains the offset within the mapped object (if any) to which the virtual address is mapped

Chapter 1. System Files
pr_pathoff
If non-zero, contains an offset into the map file to the path name and archive member name of a loaded object

pr_mflags
Specifies a bit-mask of protection and the following attribute flags:

MA_MAINEXEC
Indicates that mapping applies to the main executable in the process

MA_READ
Indicates that mapping is readable by the traced process

MA_WRITE
Indicates that mapping is writable by the traced process

MA_EXEC
Indicates that mapping is executable by the traced process

MA_SHARED
Indicates that mapping changes are shared by the mapped object

A contiguous area of the address space having the same underlying mapped object may appear as multiple mappings because of varying read, write, execute, and shared attributes. The underlying mapped object does not change over the range of a single mapping. An I/O operation to a mapping marked MA_SHARED fails if applied at a virtual address not corresponding to a valid page in the underlying mapped object. Read and write operations to private mappings always succeed. Read and write operations to unmapped addresses always fail.

cred
Contains a description of the credentials associated with the process. The file is formatted as a struct prcred type and contains the following members:

- uid_t pr_euid; /* effective user id */
- uid_t pr_ruid; /* real user id */
- uid_t pr_suid; /* saved user id (from exec) */
- gid_t pr_egid; /* effective group id */
- gid_t pr_rgid; /* real group id */
- gid_t pr_sgid; /* saved group id (from exec) */
- uint32_t pr_ngroups; /* number of supplementary groups */
- gid_t pr_groups[1]; /* array of supplementary groups */

sysent
Contains information about the system calls available to the process. The file can be used to find the number of a specific system call to trace. It can be used to find the name of a system call associated with a system call number returned in a lwpstatus file.

The file consists of a header section followed by an array of entries, each of which corresponds to a system call provided to the process. Each array entry contains the system call number and an offset into the sysent file to that system call’s null-terminated name.

The sysent file is characterized by the following attributes:

- The system call names are the actual kernel name of the exported system call.
- The entries in the array do not have any specific ordering.
- There may be gaps in the system call numbers.
- Different processes may have different system call names and numbers, especially between a 32-bit process and a 64-bit process. Do not assume that the same names or numbers cross different processes.
- The set of system calls may change during while the operating system is running. You can add system calls while the operating system is running.
- The names and numbers of the system calls may change within different releases or when service is applied to the system.
cwd  A link that provides access to the current working directory of the process. Any process can access the current working directory of the process through this link, provided it has the necessary permissions.

fd  Contains files for all open file descriptors of the process. Each entry is a decimal number corresponding to an open file descriptor in the process. If an entry refers to a regular file, it can be opened with normal file semantics. To ensure that the controlling process cannot gain greater access, there are no file access modes other than its own read/write open modes in the controlled process. Directories will be displayed as links. An attempt to open any other type of entry will fail (hence it will display 0 permission when listed).

object  A directory containing read-only files with names as they appear in the entries of the map file, corresponding to objects mapped into the address space of the target process. Opening such a file yields a descriptor for the mapped file associated with a particular address-space region. The name a.out also appears in the directory as a synonym for the executable file associated with the text of the running process.

The object directory makes it possible for a controlling process to get access to the object file and any shared libraries (and consequently the symbol tables), without the process first obtaining the specific path names of those files.

lwp  A directory containing entries each of which names a kernel thread within the containing process. The names of entries in this directory are thread ID (tid) numbers. These entries are directories containing additional files described below.

The /proc/pid/lwp/tid Structure
The directory /proc/pid/lwp/tid contains the following entries:

lwpctl  This is a write-only control file. The messages written to this file affect only the associated thread rather than the process as a whole (where appropriate).

lwpstatus  Contains thread-specific state information. This information is also present in the status file of the process for its representative thread. The file is formatted as a struct lwpstatus and contains the following members:

- `uint64_t pr_lwpid; /* specific thread id */`
- `uint32_t pr_flags; /* thread status flags */`
- `char pr_state; /* thread state - from thread.h t_state */`
- `uint16_t pr_cursig; /* current signal */`
- `uint16_t pr_why; /* reason for stop (if stopped) */`
- `uint16_t pr_what; /* more detailed reason */`
- `uint32_t pr_policy; /* scheduling policy */`
- `char pr_clname[PRCLSZ]; /* printable character representing pr_policy */`
- `pr_sigset_t pr_lwppend; /* set of signals pending to the thread */`
- `pr_sigset_t pr_lwpfhold; /* set of signals blocked by the thread */`
- `pr_siginfo64_t pr_info; /* info associated with signal or fault */`
- `pr_stack64_t pr_altstack; /* alternate signal stack info */`
- `struct pr_sigaction64 pr_action; /* signal action for current signal */`
- `uint16_t pr_syscall; /* system call number (if in syscall) */`
- `uint16_t pr_nsysarg; /* number of arguments to this syscall */`
- `uint64_t pr_sysarg[PRSYSARGS]; /* arguments to this syscall */`
- `prgregset_t pr_reg; /* general and special registers */`
- `prfpregset_t pr_ffpreg; /* floating point registers */`
- `pfamily_t pr_family; /* hardware platform specific information */`

The members of the lwpstatus file are described below:

pr_flags  Specifies a bit-mask holding these flags:
PR_STOPPED
Indicates that the thread is stopped

PR_ISTOP
Indicates that the thread is stopped on an event of interest (see the PCSTOP signal)

PR_DSTOP
Indicates that the thread has a stop directive in effect (see the PCSTOP signal)

PR_ASLLEEP
Thread is in an interruptible sleep within a system call

PR_NOREGS
No register state was provided for the thread

pr_why and pr_what
Provides the reason for why a thread was stopped. The following are possible values of the pr_why member:

PR_REQUESTED
Shows that the thread was stopped in response to a stop directive, normally because the PCSTOP signal was applied or because another thread stopped on an event of interest and the asynchronous-stop flag (see the PCSET signal) was not set for the process. The pr_what member is unused in this case.

PR_SIGNALLED
Shows that the thread stopped on receipt of a signal (see the PCSTRACE signal). The pr_what signal holds the signal number that caused the stop (for a newly-stopped thread, the same value is given with the pr_cursig member).

PR_FAULTED
Shows that the thread stopped upon incurring a hardware fault (see the PCSFAULT signal). The pr_what member contains the fault number that caused the stop.

PR_SYSENTRY
Shows a stop on entry to a system call (see the PCSENTRY signal). The pr_what member contains the system call number.

PR_SYSEXIT
Shows a stop on exit from a system call (see the PCSEXIT signal). The pr_what contains the system call number.

PR_JOBCONTROL
Shows that the thread stopped because of the default action of a job control stop signal (see the sigaction subroutine). The pr_what member contains the stopping signal number.

pr_lwpid
Names the specific thread I/O

pr_cursig
Names the current signal, which is the next signal to be delivered to the thread. When the thread is stopped by the PR_SIGNALLED or PR_FAULTED signal, the pr_info member contains additional information pertinent to the particular signal or fault. The amount of data contained in the pr_info member depends on the stop type and whether on not the application specified the SA_SIGINFO flag when the signal handler was established. For PR_FAULTED stops and PR_SIGNALLED stops when the SA_SIGINFO flag was not specified, only the si_signo, si_code, and si_addr pr_info fields contain data. For PR_SIGNALLED stops when the SA_SIGINFO flag is specified, the other pr_info fields contain data as well.
pr_action
Contains the signal action information about the current signal. This member is undefined if the pr_cursig member is zero. See the sigaction subroutine.

pr_lwppend
Identifies any synchronously generated or thread-directed signals pending for the thread. It does not include signals pending at the process level.

pr_altstack
Contains the alternate signal stack information for the thread. See the sigaltstack subroutine.

pr_syscall
Specifies the number of the system call, if any, that is being executed by the thread. It is non-zero if and only if the thread is stopped on PR_SYSENTRY or PR_SYSEXIT.

If the pr_syscall member is non-zero, the pr_nsysarg member is the number of arguments to the system call and the pr_sysarg array contains the arguments. In AIX 5.1, the pr_nsysarg member is always set to 8, the maximum number of system call parameters. The pr_sysarg member always contain eight arguments.

pr_sname
Contains the name of the scheduling class of the thread

pr_reg
Structure containing the threads general and special registers. The size and field names of this structure are machine dependent. See the <sys/m_procs.h> header file for description of this structure for your particular machine. The contents of this structure are undefined if the thread is not stopped.

pr_fpreg
Structure containing the threads floating point registers. The size and field names of this structure are machine dependent. The contents of this structure are undefined if the thread is not stopped.

pr_family
Contains the machine-specific information about the thread. Use of this field is not portable across different architectures. The pr_family structure contains extended context offset and size fields, which, if non-zero, indicate the availability of extended machine context information for the thread. A subsequent read of the status or lwpstatus file at the specified offset and size will retrieve the extended context information corresponding to a prextset structure. Alternatively, the entire lwpstatus file can be read and formatted as a struct lwpstatusx, which includes the prextset extension. The pr_family extended context offset and size members, if non-zero, indicate if the prextset member of the lwpstatusx structure is valid.

lwpsinfo
Contains information about the thread needed by the ps command. This information is also present in the psinfo file of the process for its representative thread if it has one. The file is formatted as a struct lwpsinfo type containing the following members:

```c
uint64_t pr_lwpid; /* thread id */
prptr64_t pr_addr; /* internal address of thread */
prptr64_t pr_wchan; /* wait addr for sleeping thread */
uint32_t pr_flag; /* thread flags */
uchar_t pr_wtype; /* type of thread wait */
uchar_t pr_state; /* numeric scheduling state */
char pr_sname; /* printable character representing pr_state */
uchar_t pr_nice; /* nice for cpu usage */
int pr_pri; /* priority, high value = high priority*/
uint32_t pr_policy; /* scheduling policy */
```
char     pr_clname[PRCLSZ];  /* printable character representing pr_policy */
cpu_t    pr_onpro;          /* processor on which thread last ran */
cpu_t    pr_bindpro;        /* processor to which thread is bound */

Some of the entries in the lwpsinfo file, such as pr_flag, pr_addr, pr_state, pr_wtype, and pr_wchan, refer to internal kernel data structures and should not be expected to retain their meanings across different versions of the operating system. They have no meaning to a program and are only useful for manual interpretation by a user aware of the implementation details.

Control Messages
Process state changes are effected through messages written to the ctl file of the process or to the lwpcnt file of an individual thread. All control messages consist of an int operation code identifying the specific operation followed by additional data containing operands (if any). Multiple control messages can be combined in a single write subroutine to a control file, but no partial writes are permitted. Each control message (operation code plus operands) must be presented in its entirety to the write subroutine, not in pieces through several system calls.

The following are allowed control messages:

**Note:** Writing a message to a control file for a process or thread that has already exited elicits the error ENOENT.

**PCSTOP, PCDSTOP, PCWSTOP**
When applied to the process control file,
- PCSTOP directs all threads to stop and waits for them to stop;
- PCDSTOP directs all threads to stop without waiting for them to stop;
- PCWSTOP simply waits for all threads to stop.

When applied to a thread control file,
- PCSTOP directs the specific thread to stop and waits until it has stopped;
- PCDSTOP directs the specific thread to stop without waiting for it to stop;
- PCWSTOP simply waits for the thread to stop.

When applied to a thread control file, PCSTOP and PCWSTOP complete when the thread stops on an event of interest and immediately if the thread is already stopped.

When applied to the process control file, PCSTOP and PCWSTOP complete when every thread has stopped on an event of interest.

An event of interest is either a PR_REQUESTED stop or a stop that has been specified in the process’s tracing flags (set by PCSTRACE, PCSFAULT, PCSENTRY, and PCSEXIT). PR_JOBCONTROL and PR_SUSPENDED stops are not events of interest. (A thread may stop twice because of a stop signal; first showing PR_SIGNALLED if the signal is traced and again showing PR_JOBCONTROL if the thread is set running without clearing the signal.) If PCSTOP or PCDSTOP is applied to a thread that is stopped, but not because of an event of interest, the stop directive takes effect when the thread is restarted by the competing mechanism; at that time the thread enters a PR_REQUESTED stop before executing any user-level code.

A write operation of a control message that blocks is interruptible by a signal so that, for example, an alarm subroutine can be set to avoid waiting for a process or thread that may never stop on an event of interest. If PCSTOP is interrupted, the thread stop directives remain in effect even though the write subroutine returns an error.

A kernel process (indicated by the PR_ISSYS flag) is never executed at user level and cannot be stopped. It has no user-level address space visible through the /proc file system. Applying PCSTOP, PCDSTOP, or PCWSTOP to a system process or any of its threads elicits the error EBUSY.
PCRUN
Executes a thread again after it was stopped. The operand is a set of flags, contained in an int operand, describing optional additional actions.

The allowed flags for **PCRUN** are described below:

**PRCSIG**
Clears the current signal, if any. See **PCSSIG**.

**PRCFault**
Clears the current fault, if any. See **PCCFAULT**.

**PRSTEP**
Directs the thread to execute a single machine instruction. On completion of the instruction, a trace trap occurs. If FLTTRACE is being traced, the thread stops, otherwise it is sent SIGTRAP. If SIGTRAP is being traced and not held, the thread stops. When the thread stops on an event of interest the single-step directive is cancelled, even if the stop occurs before the instruction is executed.

**PRSAAbort**
Instructs the thread to abort execution of the system call. See **PCSENTRY** and **PCSEXIT**. It is significant only if the thread is in a **PR_SYSENTRY** stop or is marked **PR_ASLEEP**.

**PRSTOP**
Directs the thread to stop again as soon as possible after resuming execution. See **PCSTOP**. In particular, if the thread is stopped on **PR_SIGNALLED** or **PR_FAULTED**, the next stop shows **PR_REQUESTED**, no other stop intervenes, and the thread does not execute any user-level code.

When applied to a thread control file, **PCRUN** makes the specific thread runnable. The operation fails and returns the error **EBUSY** if the specific thread is not stopped on an event of interest.

When applied to the process control file, a representative thread is chosen for the operation as described for the `/proc/pid/status` file. The operation fails and returns the error **EBUSY** if the representative thread is not stopped on an event of interest. If **PRSTEP** or **PRSTOP** were requested, **PCRUN** makes the representative thread runnable. Otherwise, the chosen thread is marked **PR_REQUESTED**. If as a result all threads are in the **PR_REQUESTED** stop state, they all become runnable.

Once **PCRUN** makes a thread runnable, it is no longer stopped on an event of interest, even if it remains stopped because of a competing mechanism.

**PCSTRace**
Defines a set of signals to be traced in the process. Upon receipt of one of these signals, the thread stops. The set of signals is defined using an operand **pr_sigset_t** contained in the control message. Receipt of SIGKILL cannot be traced. If you specify SIGKILL, the thread ignores it.

If a signal that is included in a thread’s held signal set is sent to that thread, the signal is not received and does not cause a stop until it is removed from the held signal set. Either the thread itself removes it or you remove it by setting the held signal set with **PCSHOLD** or the **PRSHOLD** option of **PCRUN**.

**PCSSIG**
Specifies the current signal and its associated signal information for the specific thread or representative thread. This information is set according with the operand **pr_siginfo64** structure. If the specified signal number is zero, the current signal is cleared. The error **EBUSY** is returned if the thread is not stopped on an event of interest.

The syntax of this operation are different from those of the **kill** subroutine, **pthread_kill** subroutine, or **PCKILL**. With **PCSSIG**, the signal is delivered to the thread immediately after.
execution is resumed (even if the signal is being held) and an additional PR\_SIGNALLED stop
does not intervene even if the signal is being traced. Setting the current signal to SIGKILL ends
the process immediately.

PCKILL
If applied to the process control file, a signal is sent to the process with syntax identical to those of
the kill subroutine. If applied to a thread control file, a signal is sent to the thread with syntax
identical to those of the pthread\_kill subroutine. The signal is named in an operand int
contained in the message. Sending SIGKILL ends the process or thread immediately.

PCUNKILL
Specifies a signal to be removed from the set of pending signals. If applied to the process control
file, the signal is deleted from the process’s pending signals. If applied to a thread control file, the
signal is deleted from the thread’s pending signals. The current signal (if any) is unaffected. The
signal is named in an operand int in the control message. An attempt to delete SIGKILL results in
the error EINVAL.

PCSHOLD
Sets the set of held signals for the specific or representative thread according to the operand
sigset_t structure. Held signals are those whose delivery is delayed if sent to the thread. SIGKILL
or SIGSTOP cannot be held. If specified, they are silently ignored.

PCSFAULT
Defines a set of hardware faults to be traced in the process. When incurring one of these faults, a
thread stops. The set is defined via the operand fltset_t structure. Fault names are defined in the
<sys/procfs.h> file and include the following:

Note: Some of these may not occur on all processors; other processor-specific faults may exist in
addition to those described here.

FLTILL
Illegal instruction

FLTIPRI
Privileged instruction

FLTBP
Breakpoint trap

FLTTRACE
Trace trap

FLTACCESS
Memory access fault (bus error)

FTBOUNDS
Memory bounds violation

FLTIOVF
Integer overflow

FLTIZDIV
Integer zero divide

FTFPE
Floating-point exception

FTSTACK
Unrecoverable stack fault

When not traced, a fault normally results in the posting of a signal to the thread that incurred the
fault. If a thread stops on a fault, the signal is posted to the thread when execution is resumed
unless the fault is cleared by PCCFAULT or by the PRCFAULT option of PCRUN. The pr_info
field in `/proc/pid/status` or in `/proc/pid/lwp/tid/lwpstatus` identifies the signal to be sent and contains machine-specific information about the fault.

**PCCFAULT**
Specifies the current fault, if any, to be cleared. The associated signal is not sent to the specified thread or representative thread.

**PCENTRY**
Instructs the process’s threads to stop on entry to specified system calls. The set of system calls to be traced is defined via an operand `sysset_t` structure. When entry to a system call is being traced, a thread stops after having begun the call to the system but before the system call arguments have been fetched from the thread.

If a thread is stopped on entry to a system call (PR_SYSENTRY) or when sleeping in an interruptible system call (PR_ASLEEP is set), it may be instructed to go directly to system call exit by specifying the PRSABORT flag in a PCRUN control message. Unless exit from the system call is being traced, the thread returns to user level showing error EINTR.

**PCEXIT**
Instructs the process’s threads to stop on exit from specified system calls. The set of system calls to be traced is defined via an operand `sysset_t` structure. When exit from a system call is being traced, a thread stops on completion of the system call just before checking for signals and returning to user level. At this point, all return values have been stored into the threads’s registers.

**PCSET, PCRESET, PCUNSET**
- **PCSET** sets one or more modes of operation for the traced process. **PCRESET** or **PCUNSET** resets these modes. The modes to be set or reset are specified by flags in an operand `int` in the control message:
  - **PR_FORK (inherit-on-fork)**
    When set, the tracing flags of the process are inherited by the child of a `fork` or `vfork` subroutine. When reset, child processes start with all tracing flags cleared.
  - **PR_RLC (run-on-last-close)**
    When set and the last writable /proc file descriptor referring to the traced process or any of its thread is closed, all the tracing flags of the process are cleared, any outstanding stop directives are cancelled, and if any threads are stopped on events of interest, they are set running as though PCRUN had been applied to them. When reset, the process’s tracing flags are retained and threads are not set running on last close.
  - **PR_KLC (kill-on-last-close)**
    When set and the last writable /proc file descriptor referring to the traced process or any of its threads is closed, the process is exited with SIGKILL.
  - **PR_ASYNC (asynchronous-stop)**
    When set, a stop on an event of interest by one thread does not directly affect other threads in the process. When reset and a thread stops on an event of interest other than PR_REQUESTED, all other threads in the process are directed to stop.

The error EINVAL is returned if you specify flags other than those described above or to apply these operations to a system process. The current modes are reported in the `pr_flags` field of the `/proc/pid/status` file.

**PCSREG**
Sets the general and special registers for the specific or representative thread according to the operand `prgregset_t` structure. There may be machine-specific restrictions on the allowable set of changes. PCSREG fails and returns the error EBUSY if the thread is not stopped on an event of interest or is stopped in the kernel.

**PCSFPREG**
Sets the floating point registers for the specific or representative thread according to the operand `fpregset_t` structure. The error EINVAL is returned if the system does not support floating-point
operations (no floating-point hardware and the system does not emulate floating-point machine instructions). **PCSFPREG** fails and returns the error **EBUSY** if the thread is not stopped on an event of interest or is stopped in the kernel.

**PCSVREG**
Sets the vector registers for the specific or representative thread according to the operand **prvregset_t** structure. The error **EINVAL** is returned if the system does not support vector operations (no vector hardware and the system does not emulate vector machine instructions), or if the representative thread has not referenced the vector unit. **PCSVREG** fails and returns the error **EBUSY** if the thread is not stopped on an event of interest or is stopped in the kernel.

**PCNICE**
The traced process’s **nice** priority is incremented by the amount contained in the operand **int**. Only the superuser may better a process’s priority in this way, but any user may make the priority worse. This operation is significant only when applied to a process in the time-sharing scheduling class.

### Files

<table>
<thead>
<tr>
<th>Directory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/proc</td>
<td>Directory (list of processes)</td>
</tr>
<tr>
<td>/proc/pid</td>
<td>Directory for the process <strong>pid</strong></td>
</tr>
<tr>
<td>/proc/pid/status</td>
<td>Status of process <strong>pid</strong></td>
</tr>
<tr>
<td>/proc/pid/ctl</td>
<td>Control file for process <strong>pid</strong></td>
</tr>
<tr>
<td>/proc/pid/psinfo</td>
<td>ps info for process <strong>pid</strong></td>
</tr>
<tr>
<td>/proc/pid/as</td>
<td>Address space of process <strong>pid</strong></td>
</tr>
<tr>
<td>/proc/pid/map</td>
<td>as map info for process <strong>pid</strong></td>
</tr>
<tr>
<td>/proc/pid/object</td>
<td>Directory for objects for process <strong>pid</strong></td>
</tr>
<tr>
<td>/proc/pid/sigact</td>
<td>Signal actions for process <strong>pid</strong></td>
</tr>
<tr>
<td>/proc/pid/sysent</td>
<td>System call information for process <strong>pid</strong></td>
</tr>
<tr>
<td>/proc/pid/lwp/tid</td>
<td>Directory for thread <strong>tid</strong></td>
</tr>
<tr>
<td>/proc/pid/lwp/tid/lwpstatus</td>
<td>Status of thread <strong>tid</strong></td>
</tr>
<tr>
<td>/proc/pid/lwp/tid/lwpctl</td>
<td>Control file for thread <strong>tid</strong></td>
</tr>
<tr>
<td>/proc/pid/lwp/tid/lwpinfo</td>
<td>ps info for thread <strong>tid</strong></td>
</tr>
</tbody>
</table>

### Error Codes

Other errors can occur in addition to the errors normally associated with file system access:

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENOENT</td>
<td>The traced process or thread has exited after being opened.</td>
</tr>
<tr>
<td>EFAULT</td>
<td>A read or write request was begun at an invalid virtual address.</td>
</tr>
<tr>
<td>EIO</td>
<td>A write subroutine was attempted at an illegal address in the traced process.</td>
</tr>
<tr>
<td>EBUSY</td>
<td>This error is returned because of one of the following reasons:</td>
</tr>
<tr>
<td></td>
<td>• <strong>PCSTOP, PCDSTOP</strong> or <strong>PCWSTOP</strong> was applied to a system process.</td>
</tr>
<tr>
<td></td>
<td>• An exclusive open subroutine was attempted on a process file already open for writing.</td>
</tr>
<tr>
<td></td>
<td>• <strong>PCRUN, PCSSIG, PCSREG</strong> or <strong>PCSFPREG</strong> was applied to a process or thread that was not stopped on an event of interest.</td>
</tr>
<tr>
<td></td>
<td>• An attempt was made to mount the /proc file system when it is already mounted.</td>
</tr>
</tbody>
</table>
Error Code | Description
---|---
EPERM | Someone other than the privileged user attempted to better a process's priority by issuing `PCNICE`.
ENOSYS | An attempt was made to do an unsupported operation (such as create, remove, link, or unlink) on an entry in the /proc file system.
EINVAL | An invalid argument was supplied to a system call. The following are some—but not all—possible conditions that can elicit this error:
  - A control message operation code is undefined.
  - A control message is ill-formed.
  - The PRSTEP option of the PCRUN operation was used on an implementation that does not support single-stepping.
  - An out of range signal number was specified with PCSSIG, PCKILL, or PCUNKILL.
  - SIGKILL was specified with PCUNKILL.
  - PCSFPRG was applied on a machine without floating-point support.
  - PCSVRG was applied on a machine without vector support or was applied to a thread that has not referenced the vector unit.
EINTR | A signal was received by the controlling process while waiting for the traced process or thread to stop via PCSTOP or PCWSTOP.
EBADF | The traced process performed an exec subroutine on a setuid/setgid object file or on an object file that is not readable for the process. All further operations on the process or thread file descriptor (except the close subroutine) elicit this error.

Security
The effect of a control message is guaranteed to be atomic with respect to the traced process.

For security reasons, except for the privileged user, an open subroutine of a /proc file fails unless both the effective user ID and effective group ID of the caller match those of the traced process and the process's object file is readable by the caller. Files corresponding to setuid and setgid processes can be opened only by the privileged user. Even if held by the privileged user, an open process or thread file descriptor becomes invalid if the traced process performs an exec subroutine on a setuid/setgid object file or on an object file that it cannot read. Any operation performed on an invalid file descriptor, except for the close subroutine, fails with EBADF. In this case, if any tracing flags are set and the process or any thread file is open for writing, the process is directed to stop and its run-on-last-close flag is set (see PCSET).

This feature enables a controlling process (that has the necessary permission) to reopen the process file to obtain new valid file descriptors, close the invalid file descriptors, and proceed. Just closing the invalid file descriptors causes the traced process to resume execution with no tracing flags set. Any process that is not currently open for writing with tracing flags left over from a previous open subroutine and that performs an exec subroutine on a setuid/setgid or unreadable object file is not stopped. However, that process does not have all its tracing flags cleared.
proxy.ldif.template File

Purpose
Defines the ACL that will be set for the proxy identity when the `mksecldap` command is invoked with the `-x` and `-X` command options.

Description
The `proxy.ldif.template` file contains LDAP data interchange formatted (LDIF) entries used by `mksecldap` when creating a proxy identity during server setup. By default, the file contains entries to create the proxy identity and password and set the default ACL to propagate down from the base DN (distinguished name).

Entries in the file may be modified or added by the system administrator to customize the LDAP server setup performed by the `mksecldap` command. Several case sensitive key words exist in the file that are dynamically replaced with the values that `mksecldap` is invoked with as described in the following table.

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Substitution</th>
</tr>
</thead>
<tbody>
<tr>
<td>{baseDN}</td>
<td>Replaced with base distinguished name specified by the <code>-d</code> option of <code>mksecldap</code> command.</td>
</tr>
<tr>
<td>(proxyDN)</td>
<td>Replaced with proxy user distinguished name specified by the <code>-x</code> option of <code>mksecldap</code>.</td>
</tr>
<tr>
<td>(proxyUser)</td>
<td>Replaced with proxy user name (proxyDN stripped of suffix and prefix).</td>
</tr>
<tr>
<td>(proxyPWD)</td>
<td>Replaced with proxy user password specified by the <code>-X</code> option of <code>mksecldap</code>.</td>
</tr>
</tbody>
</table>

Related Information
The `mksecldap` command.


pwdhist File

Purpose
Contains password history information.

Description
The `/etc/security/pwdhist.dir` and `/etc/security/pwdhist.pag` files are database files created and maintained by Database Manager (DBM) subroutines. The files maintain a list of previous user passwords.

The `pwdhist` files store information by user name. User names are the keys of the DBM subroutines. The password list contains multiple pairs of a `lastupdate` value and an encrypted, null-terminated password. This password list is a key’s associated content and the `lastupdate` value is a 4-byte, unsigned long. The encrypted password is the size of the `PW_CRYPTLEN` value. Thus, an entry in the database file is of the following format:

```
lastupdatepasswordlastupdatepasswordlastupdatepassword...
```

The password list is in descending chronological order, with the most recent password appearing first in the list.

To retrieve a user's password history, use the `dbm_fetch` subroutine. To delete a user's password history, use the `dbm_delete` subroutine.
Security
Access Control: The files grant read and write access only to the root user.

Examples
If user sally has the following previous passwords:

password = 6PugcayXL.1Rw ; lastupdate = 737161212
password = r5MZvr69mGeLE ;
lastupdate = 746458629

the dbm_fetch subroutine returns the following entry for the key sally:
746458629r5MZvr69mGeLE7371612126PugcayXL.1Rw

Related Information
The /etc/security/passwd file, /etc/security/user file.
The passwd command.

For lists of DBM and NDBM Subroutines, see List of NDBM and DBM Programming References in AIX 5L Version 5.3 Communications Programming Concepts.

publickey File for NIS

Purpose
Contains public or secret keys for maps.

Description
The /etc/publickey file is the public key file used for secure networking. Each entry in the file consists of a network user name (which may refer to either a user or a host name), followed by the user's public key (in hex notation), a colon, and then the user’s secret key encrypted with its login password (also in hex notation). This file is part of the Network Support Facilities.

This file is altered either by the user through the chkey command or by the person who administers the system through the newkey command. The publickey file should only contain data on the master server, where it is converted into the publickey.byname NIS map.

Related Information
The chkey command, keylogin command, newkey command.
The keyserv daemon, ypupdated daemon.


qconfig File

Purpose
Configures a printer queuing system.
Description

The `/etc/qconfig` file describes the queues and devices available for use by both the `enq` command, which places requests on a queue, and the `qdaemon` command, which removes requests from the queue and processes them. The `qconfig` file is an attribute file.

Some stanzas in this file describe queues, and other stanzas describe devices. Every queue stanza requires that one or more device stanzas immediately follow it in the file. The first queue stanza describes the default queue. Unless the `LPDEST` or `PRINTER` environment variable is set, the `enq` command uses this queue when it receives no queue parameter. If `LPDEST` contains a value, that value takes precedence over the `PRINTER` environment variable. Destination command-line options always override both variables.

The name of a queue stanza can be from 1 to 20 characters long. Some of the fields and their possible values that can appear in this file are:

<table>
<thead>
<tr>
<th>Field</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>acctfile</td>
<td>Identifies the file used to save print accounting information. <strong>FALSE</strong>, the default value, indicates suppress accounting. If the named file does not exist, no accounting is done.</td>
</tr>
<tr>
<td>device</td>
<td>Identifies the symbolic name that refers to the device stanza.</td>
</tr>
<tr>
<td>discipline</td>
<td>Defines the queue serving algorithm. <strong>fcfs</strong>, means first-come-first-served. <strong>sjn</strong> means shortest job next.</td>
</tr>
<tr>
<td>up</td>
<td>Defines the state of the queue. <strong>TRUE</strong>, the default value, indicates that the queue is running. <strong>FALSE</strong> indicates that it is not running.</td>
</tr>
<tr>
<td>recovery_type</td>
<td>Enables users to specify a recovery option when a print queue goes down. By default, the queue remains in the down state pending user intervention.</td>
</tr>
</tbody>
</table>

Other options can be specified using the following values:

- **runscript** `<PathName>`
  A user-defined script is run when the queue goes down. The actions taken by the script are left to the discretion of the system administrator.

- **retry** `-T <delay>` `-R <retries>`
  The queue will stay down for the period of time specified in `delay` (expressed in minutes). Subsequently, it will be taken back up and the job will be retried. This is repeated up to the number of retries specified in `retries`. This is particularly useful in cases where job failures are due to temporary conditions likely to be resolved within the lapse of a certain time period (for example, a paper out condition or a temporary network glitch or slowdown).

- **sendmail** `<username>`
  The specified user will receive mail when the queue goes down notifying him or her that the specific printer is down.

Note: **lp** is a BSD standard reserved queue name and should not be used as a queue name in the `qconfig` file.

The following list shows some of the fields and their possible values that appear in the `qconfig` file for remote queues:

- **host** Indicates the remote host where the remote queue is found.
s_statfilter Specifies the short version filter used to translate remote queue status format. The following are possible values:

- /usr/lib/lpd/bsdshort
  BSD remote system
- /usr/lib/lpd/aixv2short
  RT remote system
- /usr/lib/lpd/attshort
  AT&T remote system

l_statfilter Specifies the long version filter used to translate remote queue status format. The following are possible values:

- /usr/lib/lpd/bsdlong
  BSD remote system
- /usr/lib/lpd/aixv2long
  RT remote system
- /usr/lib/lpd/attlong
  AT&T remote system

rq Specifies the remote queue name. In a remote print environment, the client configuration should specify the remote queue name or the server. Using the default remote queue name may cause unpredictable results.

If a field is omitted, its default value is assumed. The default values for a queue stanza are:

discipline = fcfs
up = TRUE
acctfile = FALSE
recovery_type = queuedown

The device field cannot be omitted.

The name of a device stanza is arbitrary and can be from 1 to 20 characters long. The fields that can appear in the stanza are:

<table>
<thead>
<tr>
<th>Field</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>access</td>
<td>Specifies the type of access the backend has to the file specified by the file field. The value of access is write if the backend has write access to the file or both if it has both read and write access. This field is ignored if the file field has the value FALSE.</td>
</tr>
<tr>
<td>align</td>
<td>Specifies whether the backend sends a form-feed control before starting the job if the printer was idle. The default value is TRUE.</td>
</tr>
<tr>
<td>backend</td>
<td>Specifies the full path name of the backend, optionally followed by the flags and parameters to be passed to it. The path names most commonly used are /usr/lib/lpd/piobe for local print and /usr/lib/lpd/rembak for remote print.</td>
</tr>
<tr>
<td>feed</td>
<td>Specifies either the number of separator pages to print when the device becomes idle or the value never, the default, which indicates that the backend is not to print separator pages.</td>
</tr>
<tr>
<td>file</td>
<td>Identifies the special file where the output of backend is to be redirected. FALSE, the default value, indicates no redirection and that the file name is /dev/null. In this case, the backend opens the output file.</td>
</tr>
<tr>
<td>header</td>
<td>Specifies whether a header page prints before each job or group of jobs. A value of never, the default value, indicates no header page at all. always means a header page before each job. group means a header before each group of jobs for the same user. In a remote print environment, the default action is to print a header page and not to print a trailer page.</td>
</tr>
<tr>
<td>trailer</td>
<td>Specifies whether a trailer page prints after each job or group of jobs. A value of never, the default, means no trailer page at all. always means a trailer page after each job. group means a trailer page after each group of jobs for the same user. In a remote print environment, the default action is to print a header page and not to print a trailer page.</td>
</tr>
</tbody>
</table>
The `qdaemon` process places the information contained in the feed, header, trailer, and align fields into a status file that is sent to the backend. Backends that do not update the status file do not use the information it contains.

If a field is omitted, its default value is assumed. The backend field cannot be omitted. The default values in a device stanza are:

- `file` = FALSE
- `access` = write
- `feed` = never
- `header` = never
- `trailer` = never
- `align` = TRUE

The `enq` command automatically converts the ASCII `qconfig` file to binary format when the binary version is missing or older than the ASCII version. The binary version is found in the `/etc/qconfig.bin` file.

**Note:** The `qconfig` file should not be edited while there are active jobs in any queue. Any time the `qconfig` file is changed, jobs submitted prior to the change will be processed before jobs submitted after the change.

Editing includes both manual editing and use of the `mkque`, `rmque`, `chque`, `mkquedev`, `rmquedev`, or `chquedev` command. It is recommended that all changes to the `qconfig` file be made using these commands. However, if manual editing is desired, first issue the `enq -G` command to bring the queuing system and the `qdaemon` to a halt after all jobs are processed. Then edit the `qconfig` file and restart the `qdaemon` with the new configuration.

**Examples**

1. The batch queue supplied with the system might contain these stanzas:

   ```
   bsh:
   discipline = fcfs
   device = bshdev
   bshdev:
   backend = /usr/bin/ksh
   
   bsh1:
   backend = /usr/bin/ksh
   bsh2:
   backend = /usr/bin/ksh
   
   To run a shell procedure called `myproc` using this batch queue, enter:
   qprt -Pbsh myproc
   
   The queuing system runs the files one at a time, in the order submitted. The `qdaemon` process redirects standard input, standard output, and standard error to the `/dev/null` file.
   ```

2. To allow two batch jobs to run at once, enter:

   ```
   bsh:
   discipline = fcfs
   device = bsh1,bsh2
   bsh1:
   backend = /usr/bin/ksh
   bsh2:
   backend = /usr/bin/ksh
   
   To set up a remote queue, `bsh`, enter:
   ```

3. To set up a remote queue, `bsh`, enter:

   ```
   remh:
   device = rd0
   host = pluto
   rq = bsh
   rd0:
   backend = /usr/lib/lpd/rembak
   ```

4. To set a local queue such that mail is sent to `user1@xyz.com` when it goes down, enter:

   ```
   ps:
   recovery_type = sendmail user1@xyz.com
   device = lp0
   ```
lp0:
  file = /dev/lp0
  header = never
  trailer = never
  access = both
  backend = /usr/lib/lpd/piobe

Files

/etc/qconfig
/etc/qconfig.bin
/dev/null
/usr/lib/lpd/piobe
/usr/lib/lpd/rembak
/usr/lib/lpd/digest

Contains the configuration file.
Contains the digested, binary version of the /etc/qconfig file.
Provides access to the null device.
Specifies the path of the local printer backend.
Specifies the path of the remote printer backend.
Contains the program that converts the /etc/qconfig file to binary format.

Related Information

The enq command, lp command, qdaemon command.

Backend and qdaemon interaction in AIX 5L Version 5.3 General Programming Concepts: Writing and Debugging Programs.

Printing administration and Print spooler in Printers and printing.

rc.boot File

Purpose
Controls the machine boot process.

Description
Attention: Executing the rc.boot script on a system that is already running may cause unpredictable results.

The /sbin/rc.boot file is a shell script that is called by the simple shell init and the standard init command to bring up a system. Depending upon the type of boot device, the rc.boot file configures devices and also calls the appropriate applications. Appropriate applications include:

• Booting from disk
• Varying on a root volume group
• Enabling file systems
• Calling the BOS installation programs or diagnostics

The rc.boot program is only called by an init process.

Files

/etc/inittab
/usr/lib/boot/ssh

Controls the initialization process.
Calls the rc.boot file.

Chapter 1. System Files 213
Related Information

- Systems that will not boot in the Operating system and device management.
- Logical volume storage in the Operating system and device management.

rc.tcpip File for TCP/IP

Purpose

Initializes daemons at each system restart.

Description

The `/etc/rc.tcpip` file is a shell script that, when executed, uses SRC commands to initialize selected daemons. The `rc.tcpip` shell script is automatically executed with each system restart. It can also be executed at any time from the command line.

Most of the daemons that can be initialized by the `rc.tcpip` file are specific to TCP/IP. These daemons are:

- `inetd` (started by default)
- `gated`
- `routed`
- `named`
- `timed`
- `rwhod`

**Note:** Running the `gated` and `routed` daemons at the same time on a host may cause unpredictable results.

There are also daemons specific to the base operating system or to other applications that can be started through the `rc.tcpip` file. These daemons are:

- `lpd`
- `portmap`
- `sendmail`
- `syslogd`

The `syslogd` daemon is started by default.

Examples

1. The following stanza starts the `syslogd` daemon:
   ```
   # Start up syslog daemon (for error
   or event logging)
   start /usr/sbin/syslogd "$src_running"
   ```
2. The following stanza starts the `lpd` daemon:
   ```
   # Start up print daemon
   start /usr/sbin/lpd "$src_running"
   ```
3. The following stanza starts the `routed` daemon, but not the `gated` daemon:
   ```
   # Start up routing daemon (only single
   start /usr/sbin/routed "$src_running" -g
   ```
   ```
   # start /usr/sbin/gated "$src_running"
   ```
Related Information
The `startsrc` command, `stopsrc` command.

The `gated` daemon, `inetd` daemon, `lpd` daemon, `named` daemon, `portmap` daemon, `routed` daemon, `rwhod` daemon, `sendmail` daemon, `syslogd` daemon, `timed` daemon.

TCP/IP name resolution in *Networks and communication management*.

Installation of TCP/IP in *Networks and communication management*.

---

**realm.map File**

**Purpose**
Contains the NFS local realm-to-domain maps.

**Description**
The `/etc/nfs/realm.map` file contains the local NFS realm-to-domain mappings. These mappings are used by NFS V4 to convert Kerberos principals to UNIX credentials. The `/etc/nfs/realm.map` file may be modified using the `chnfsrtd` command. The format of the `/etc/nfs/realm.map` file has one Kerberos realm and its corresponding NFS domain per line. A Kerberos realm should only be specified once in the file.

**Files**
- `/etc/nfs/realm.map` The `realm.map` file.

**Related Information**
The `chnfsrtd` command.

---

**remote.unknown File for BNU**

**Purpose**
Logs access attempts by unknown remote systems.

**Description**
The `/usr/sbin/uucp/remote.unknown` file is a shell script. It is executed by the Basic Networking Utilities (BNU) program when a remote computer that is not listed in the local `/etc/uucp/Permissions` file attempts to communicate with that local system. The BNU program does not permit the unknown remote system to connect with the local system. Instead, the `remote.unknown` shell procedure appends an entry to the `/var/spool/uucp/.Admin/Foreign` file.

Modify the `remote.unknown` file to fit the needs of your site. For example, to allow unknown systems to contact your system, remove the execute permissions for the `remote.unknown` file. You can also modify the shell script to send mail to the BNU administrator or to recognize certain systems and reject others.

**Note:** Only someone with root user authority can edit the `remote.unknown` file, which is owned by the `uucp` program login ID.

**Files**
- `/usr/sbin/uucp/remote.unknown` Contains the `remote.unknown` shell script.
- `/etc/sbin/Permissions` Describes access permissions for remote systems.
Reserved for future use.

Related Information

- \texttt{Basic Networking Utilities} in \textit{Networks and communication management}.

\section*{resource\_data\_input Information}

\subsection*{Purpose}

Provides information about using an input file for passing resource class and resource attribute names and values to the resource monitoring and control (RMC) command-line interface (CLI).

\subsection*{Description}

You can specify the name of a resource data input file with the \texttt{-f} command-line flag to pass resource persistent attribute values to the RMC CLI when using the command line directly would be too cumbersome or too prone to typographical errors. The data in this file is used for defining resources or for changing the persistent attribute values of a resource or resource class. This file has no set location. It can be a temporary file or a permanent file, depending on requirements.

The \texttt{mkrsrc} and \texttt{chrsrc} commands read this file when they are issued with the \texttt{-f} flag. The \texttt{lsrcrsrcdef} and \texttt{lsrcrsrcdef} commands generate a file with this format when issued with the \texttt{-f} flag.

\textbf{PersistentResourceAttributes}

Persistent attribute names and values for one or more resources for a specific resource class used to define a new resource or change attribute values for an existing resource. The persistent resource attributes are read in by the commands \texttt{mkrsrc} and \texttt{chrsrc}. These attributes are ignored if the input file is read by the \texttt{chrsrc} command that has been specified with the \texttt{-c} flag.

\textbf{PersistentResourceClassAttributes}

Persistent attribute names and values for a resource class used to change the attribute values of an existing resource class. The persistent resource class attributes are read in by the command \texttt{chrsrc} only when the \texttt{-c} flag is specified.

In general, a resource\_data\_input file is a flat text file with the following format. \textbf{Bold} words are literal. Text that precedes a single colon (:) is an arbitrary label and can be any alphanumeric text.

\texttt{PersistentResourceAttributes::

# This is a comment

label:
    AttrName1  = value
    AttrName2  = value
    AttrName3  = value
another label:
    Name     = name
    NodeNumber = 1

::

::

PersistentResourceClassAttributes::

label:
    SomeSettableAttrName = value
    SomeOtherSettableAttrName = value

::

::

See the \textbf{Examples} section for more details.
Some notes about formatting follow:

- The keywords **PersistentResourceAttributes** and **PersistentResourceClassAttributes** are followed by a double colon (::).
- The order of the keyword stanzas is not significant in the file. For example, **PersistentResourceClassAttributes** could precede **PersistentResourceClass**. It does not affect the portion of the data that is read in by the calling CLI.
- Individual stanza headings (beneath the keywords) are followed by a single colon (:), for example: `c175n05 resource info`.
- White space at the beginning of lines is not significant. Tabs or spaces are suggested for readability.
- Any line with a pound sign (#) as the first printable character is is a comment.
- Each entry on an individual line is separated by white space (spaces or tabs).
- Blank lines in the file are not significant and are suggested for readability.
- There is no limit to the number of resource attribute stanzas included in a particular **PersistentResourceAttributes** section.
- There is no limit to the number of resource class attribute stanzas included in a particular **PersistentResourceClassAttributes** section. Typically, there is only one instance of a resource class. In this case, only one stanza is expected.
- If only one resource attribute stanza is included in a particular **PersistentResourceAttributes** section, the `label:` line can be omitted.
- If only one resource class attribute stanza is included in a particular **PersistentResourceClassAttributes** section, the `label:` line can be omitted.
- Values that contain spaces must be enclosed in quotation marks.
- A double colon (::) indicates the end of a section. If a terminating double colon is not found, the next **Reserved Keyword** or **end of file** signals the end of a section.
- Double quotation marks included within a string that is surrounded by double quotation marks must be escaped. ('').

**Note:** Double quotation marks can be nested within single quotation marks. These are examples:
- "Name == \"testing\"
- 'Name == "testing"
  This syntax is preferred if your string is a selection string and you are going to cut and paste to the command line.

**Note:** Single quotation marks can be nested within double quotation marks. These are examples:
- 'Isn\'t that true'
- "Isn't that true"
  This syntax is preferred if you are going to cut and paste to the command line.

- The format you use to enter data in a **resource_data_input** file may not be the same format used on the command line. The shell you choose to run the commands in has its own rules with regard to quotation marks. Refer to the documentation for your shell for these rules, which determine how to enter data on the command line.

**Examples**

1. This sample **mkrsrc** command:
   ```bash```
mkrsrc -f /tmp/my_resource_data_input_file IBM.Foo
```bash```
uses the sample input file `/tmp/my_resource_data_input_file` for the `IBM.Foo` resource class. The contents of the input file look like this:

```
PersistentResourceAttributes::
# Resource 1 - only set required attributes
resource 1:
    Name="c175n04"
    NodeList = [1]
# Resource 2 - setting both required and optional attributes
# mkrsrc -e2 IBM.Foo displays required and optional
# persistent attributes
resource 2:
    Name="c175n05"
    NodeList = [1]
    Int32 = -99
    Int64 = -123456789123456789
    Float32 = -9.89
    Float64 = 123456789.123456789
    String = "testing 123"
    Binary = 0xaabbccddeeff
    RH = "0x0000 0x0000 0x00000000 0x00000000 0x00000000 0x00000000"
    SD = [hello,1,{2,4,6,8}]
    Int32Array = {-4, -3, -2, -1, 0, 1, 2, 3, 4}
    Int64Array = {-4,-3,-2,-1,0,1,2,3,4}
    Uint32Array = {0,1,2,3,4,5,6}
    Uint64Array = {0,1,2,3,4,5,6}
    Float32Array = {-3.3,-2.2,-1.2,0,1,2,2.2,3.3}
    Float64Array = {-3.3,-2.2,-1.2,0,1,2,2.2,3.3}
    StringArray = {"abc","do re mi","123"}
    BinaryArray = {"0x01","0x02","0x0304"}
    RHArray ={"0x0000 0x0000 0x00000000 0x00000000 0x00000000 0x00000000", "0xaaab 0xaaab 0xbbbbbbb 0xccccccc 0xdddddddd 0xeeeeeee"}
    SDArray = {
        [hello,1,{0,1,2,3}],[hello2,2,{2,4,6,8}]}
```

2. This sample `chsrc` command:

```
chsrc -f /tmp/Foo/ch_resources -s 'Name = "c175n05"' IBM.Foo
```

uses the sample input file `/tmp/Foo/ch_resources` to change the attribute values of existing `IBM.Foo` resources. The contents of the input file look like this:

```
PersistentResourceAttributes::
# Changing resources that match the selection string entered
# when running chsrc command.
resource 1:
    String = "this is a string test"
    Int32Array = {10,-20,30,-40,50,-60}
```

Related Information

Commands: `chsrc`, `lsactdef`, `lsrsrcdef`, `mkrsrc`

Information Files: rmccli

rmccli Information

Purpose
Provides general information about resource monitoring and control (RMC) and related commands.
Description
Provides general information about RMC and related commands, including datatypes, terminology, and references to related information.

Command structure and use
The RMC commands may be grouped into categories representing the different operations that can be performed on resource classes and resources:
- Creating and removing resources: mkrsrc, rmrsrc
- Modifying resources: chrsrc, refrsrc
- Viewing definitions and data: lsrsrc, lsrsrcdef
- Viewing actions: lsactdef
- Running actions: runact

The RMC commands can be run directly from the command line or called by user-written scripts. In addition, the RMC commands are used as the basis for higher-level commands, such as the event response resource manager (ERRM) commands.

Data display information
The flags that control the display function for the RMC CLI routines, in order of precedence, are:
1. –l for long display. This is the default display format.
   For example, the command:
   lsrsrc -s 'Name == "c175n05"' IBM.Foo Name NodeList SD Binary RH Int32Array
   produces output that looks like this:
   Persistent Attributes for Resource: IBM.Foo
   resource 1:
   Name = "c175n05"
   NodeList = [1]
   SD = ["testing 1 2 3",1,\{0,1,2\}]
   Binary = "0xaabbcc00 0xeeff"
   RH = "0x0000 0x0000 0x00000000 0x00000000 0x00000000"
   Int32Array = [1,5,-10,1000000]

2. –t for tabular display.
   For example, the command:
   lsrsrc -s 'Name == "Page"' -t IBM.Condition Name EventExpression
   produces output that looks like this:
   Persistent Attributes for Resource: IBM.Condition
   Name                  EventExpression
   "Page space out rate" "VMPgSpOutRate > 500"
   "Page fault rate" "VMPgFaultRate > 500"
   "Page out rate" "VMPgOutRate > 500"
   "Page in rate" "VMPgInRate > 500"
   "Page space in rate" "VMPgSpInRate > 500"

3. –x for suppressing headers when printing.
4. –d for colon (:) delimited display.
   For example, the command:
   lsrsrc -xd -s 'Name == "c175n05"' IBM.Foo Name Int32 Uint32Array SD Binary
   produces output that looks like this:
   c175n05:-100:1:["hel 101",1,\{0,1,2\}]:"0xaabbcc00 0xeeff":
   Note the use of the –x flag along with the –d flag.
5. –D delimiter for string-delimited display.
For example, the command:

```
lsrsrc -xD:: -s 'Name == "c175n05"' IBM.Foo Name Int32 Uint32Array SD Binary
```

produces output that looks like this:

```
c175n05::-100::{}::['hello1',1,[0,1,2]]::"0xaabbcc00 0xeeff"
```

Note the use of the `–x` flag along with the `–D Delimiter` flag.

When output of any list command (lsrsrc, lsrsrcdef) is displayed in the tabular output format, the printing column width may be truncated. If more characters need to be displayed (as in the case of strings) use the `–l` flag to display the entire field.

### Data input formatting

Binary data can be entered in the following formats:

- "0x######## 0x######## 0x####...
- "0x###################...
- 0x################...

Be careful when you specify strings as input data:

- Strings that contain no white space or non-alphanumeric characters can be entered as input without enclosing quotation marks
- Strings that contain white space or other alphanumeric characters must be enclosed in quotation marks
- Strings that contain single quotation marks (') must be enclosed by double quotation marks ("), as shown in this example: "this is a string with 'single quotations marks'"

Selection strings must be enclosed in double quotation marks, unless the selection string itself contains double quotation marks, in which case the selection string must be enclosed in single quotation marks. For information on how to specify selection strings, see the *RSCT: Administration Guide*.

- Sample selection string input: "NodeNumber == 1"
- Selection string input where double quotation marks are part of the selection string: 'Name == "c175n05"

Structured data (SD) types must be enclosed in square brackets: [hello,1,[2,4,6,8]]

When supplying structured data (SD) as command-line input to the RMC commands, enclose the SD in single quotation marks: SD='[hello,1,[2,4,6,8]]'

Arrays of any type must be enclosed in braces {}:

- Array of integers: {-4, -3, -2, -1, 0, 1, 2, 3, 4}
- Array of strings: {abc, "do re mi", 123}
- Array of structured data: {[hello,1,{0,1,2,3}],[hello2,2,{2,4,6,8}]}

Arrays of any type with more than one element must be enclosed in quotation marks. For example:

- mkrsrsrc IBM.Foo Name=testing NodeList={1} Uint32Array='{1,2,3}'
- mkrsrsrc IBM.Foo Name=testing NodeList={1} Uint32_array='{1,2,3}'

Arrays of strings and arrays of structured data must always be enclosed in quotation marks.

When supplying arrays of structured data or arrays containing strings enclosed in quotation marks as command-line input to the RMC commands, enclose the entire array in single quotation marks:
• Array of strings: mkrsrc IBM.Foo Name="c175n05" NodeList={1} StringArray='{"a string","a different string"}'

• Array of structured data: mkrsrc IBM.Foo Name="c175n05" NodeList={1} SDArrary='["string 1",1,{},1],["string 2",2,{},2,3}]

For more examples, see the resource_data_input information file.

Data output formatting
String data is always displayed in either double or single quotation marks, as shown below:

• A description attribute that equals the string "This is a string that contains white space" is displayed using long format as:
  Description = "This is a string that contains white space"

• A description attribute value that equals an empty string "" is displayed in long format as:
  Description = ""

• A description attribute value that equals a string that contains a new-line character at the end of the string is displayed in long format as:
  Description = "This string ends with a new-line character..."

• A selection string containing double quotation marks is displayed in long format as:
  SelectionString = 'Name == "c175n05"

• A name attribute value that equals the string "c175n05" is displayed in long format as:
  Name = "c175n05"

Binary data is displayed as follows:
"0x######## 0x######## 0x######## 0x###..."

Naming conventions
The following variable names are used throughout the RMC command informations:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>attr</td>
<td>The name of a resource class or a resource attribute</td>
</tr>
<tr>
<td>resource_class</td>
<td>The name of a resource class</td>
</tr>
</tbody>
</table>

Terminology

attribute
Attributes are either persistent or dynamic. A resource class is defined by a set of persistent and dynamic attributes. A resource is also defined by a set of persistent and dynamic attributes. Persistent attributes define the configuration of the resource class and resource. Dynamic attributes define a state or a performance-related aspect of the resource class and resource. In the same resource class or resource, a given attribute name can be specified as either persistent or dynamic, but not both.

resource
An entity in the system that provides a set of services. Examples of hardware entities are processors, disk drives, memory, and adapters. Examples of software entities are database applications, processes, and file systems. Each resource in the system has one or more attributes that define the state of the resource.

resource class
A broad category of system resource, for example: node, file system, adapter. Each resource class has a container that holds the functions, information, dynamic attributes, and conditions that apply to that resource class. For example, the "/tmp space used" condition applies to a file system resource class.

resource manager
A process that maps resource and resource-class abstractions into calls and commands
for one or more specific types of resources. A resource manager can be a standalone
daemon, or it can be integrated into an application or a subsystem directly.

To see all of the resource classes defined in the system, run the **lsrsrc** command without
any flags or parameters. To see all of the resources defined in the system for the
**IBM.FileSystem** resource class, enter:

```
lsrsrc IBM.FileSystem
```

**selection string**

Must be enclosed within either double or single quotation marks. If the selection string
contains double quotation marks, enclose the entire selection string in single quotation
marks, for example:

```
-s 'Name == "testing"'
```

```
-s 'Name != "test"'
```

Only persistent attributes can be listed in a selection string. For information on how to
specify selection strings, see the *RSCT: Administration Guide*.

**Flags**

- **-h**  Writes the command’s usage statement to standard output.
- **-T**  Writes the command’s trace messages to standard error. For your software service organization’s
  use only.
- **-V**  Writes the command’s verbose messages to standard output.

All RMC commands include a **-T** flag and a **-V** flag. Use the **-T** flag only when your software service
organization instructs you to turn tracing on. Trace messages are not translated. Use the **-V** flag, which
indicates “verbose” mode, to see more information about the command. Verbose messages are contained
in message catalogs and are translated based on the locale in which you are running and other criteria.

**Environment Variables**

**CT_CONTACT**

When the **CT_CONTACT** environment variable is set to a host name or IP address, the command
contacts the resource monitoring and control (RMC) daemon on the specified host. If the
environment variable is not set, the command contacts the RMC daemon on the local system
where the command is being run. The resource class or resources that are displayed or modified
by the command are located on the system to which the connection is established.

**CT_MANAGEMENT_SCOPE**

Determines the management scope that is used for the session with the RMC daemon to monitor
and control the resources and resource classes. The management scope determines the set of
possible target nodes where the resources and resource classes can be monitored and controlled.
The valid values are:

```
0  Specifies local scope.
1  Specifies local scope.
2  Specifies peer domain scope.
3  Specifies management domain scope.
```

If this environment variable is *not* set, local scope is used.
Related Information
Books: RSCT: Administration Guide, for information about RMC operations

Information Files: resource_data_input

roles File

Purpose
Contains the list of valid roles. This system file only applies to AIX 4.2.1 and later.

Description
The /etc/security/roles file contains the list of valid roles. This is an ASCII file that contains a stanza for each system role. Each stanza is identified by a role name followed by a : (colon) and contains attributes in the form Attribute=Value. Each attribute pair ends with a newline character as does each stanza.

The file supports a default stanza. If an attribute is not defined, the default value for the attribute is used.

A stanza contains the following attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rolelist</td>
<td>Contains a list of roles implied by this role and allows a role to function as a super-role. If the rolelist attribute contains the value of &quot;role1,role2&quot;, assigning the role to a user also assigns the roles of role1 and role2 to that user.</td>
</tr>
<tr>
<td>authorizations</td>
<td>Contains the list of additional authorizations acquired by the user for this specific role.</td>
</tr>
<tr>
<td>groups</td>
<td>Contains the list of groups that a user should belong to in order to effectively use this role. The user must be added to each group in this list for this role to be effective.</td>
</tr>
<tr>
<td>screens</td>
<td>Contains a list of SMIT screen identifiers that allow a role to be mapped to various SMIT screens. The default value for this attribute is * (all screens).</td>
</tr>
<tr>
<td>msgcat</td>
<td>Contains the file name of the message catalog that contains the one-line descriptions of system roles.</td>
</tr>
<tr>
<td>msgnum</td>
<td>Contains the message ID that retrieves this role description from the message catalog.</td>
</tr>
</tbody>
</table>

For a typical stanza, see the Examples stanza.

Changing the roles File
You should access this file through the commands and subroutines defined for this purpose. You can use the following commands to change the roles file:

- chrole
- lsrole
- mkrole
- rmrole

The mkrole command creates an entry for each new role in the /etc/security/roles file. To change the attribute values, use the chrole command. To display the attributes and their values, use the lsrole command. To remove a role, use the rmrole command.

To write programs that affect attributes in the /etc/security/roles file, use the subroutines listed in Related Information.
Security
Access Control: This file grants read and write access to the root user, and read access to members of the security group.

Examples
A typical stanza looks like the following example for the ManageAllUsers role:

ManageAllUsers:
  rolelist = ManageBasicUsers
  authorizations = UserAdmin,RoleAdmin,PasswdAdmin,GroupAdmin
  groups = security
  screens = mkuser,rmuser,!tcpip

Files
/etc/security/roles
Contains the list of valid roles.
/etc/security/user.roles
Contains the list of roles for each user.
/etc/security/smitacl.group
Contains the group ACL definitions.
/etc/security/smitacl.user
Contains the user ACL definitions.

Related Information
The `chrole` command, `lsrole` command, `mkrole` command, `rmrole` command.

The `getroleattr` subroutine, `nextrole` subroutine, `putroleattr` subroutine, `setroledb` subroutine, `endroledb` subroutine.

rpc File for NFS

Purpose
Contains the database for Remote Procedure Calls (RPC) program numbers using NFS.

Description
The /etc/rpc file, part the Network Support Facilities, contains names that are used in place of RPC program numbers. These names can be read by users. Each line of the file contains the following entries:

<table>
<thead>
<tr>
<th>Entry</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name of Server for the RPC Program</td>
<td>Specifies the name of the server daemon that provides the RPC program.</td>
</tr>
<tr>
<td>RPC Program Number</td>
<td>Specifies the number assigned to the program by the RPC protocol.</td>
</tr>
<tr>
<td>Aliases</td>
<td>Specifies alternate names by which the service can be requested.</td>
</tr>
</tbody>
</table>

The three entries for each line are entered in the order listed here. Entries can be separated by any number of blanks or tab characters, provided the line does not wrap. Commented lines in the file must begin with a # (pound sign). Characters in a commented line are not interpreted by routines that search the file.

Examples
A sample /etc/rpc file follows:
Related Information

File systems in Operating system and device management.

sectoldif.cfg Configuration File

Purpose
 Defines the names to use for defined data types when generating directory information tree (DIT) data for LDAP.

Description
 The sectoldif.cfg configuration file is used by the mksecldap, sectoldif, and nistoldif commands when generating output to export to LDAP. This file allows a system administrator to customize the naming used for various data branches that will be created in LDAP. Default names are provided and may be used unless customization is desired. Each entry in the file consist of the following fields:

<table>
<thead>
<tr>
<th>Data_Type</th>
<th>LDAP_Attribute_Name</th>
<th>LDAP_Object_Class</th>
<th>LDAP_Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specifies the data type. Values are USER, GROUP, ID, HOST, SERVICE, PROTOCOL, NETWORK, NETGROUP, and RPC.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specifies the LDAP attribute name.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specifies the LDAP object class associated with LDAP_Attribute_Name.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specifies the LDAP attribute value.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The Data_Type field must be one of the recognized types. The remaining fields in an entry are configurable by the system administrator. System administrator must ensure that the LDAP_Object_Class field is appropriate for the supplied LDAP_Attribute_Name as the commands do not verify the combination.

Examples
 1. The following modifications to sectoldif.cfg will cause users to be exported to ou=Employees,o=ibm and groups to ou=Departments,o=ibm when -d o=ibm is specified for the sectoldif command:

     USER ou organizationalUnit Employees
     GROUP ou organizationalUnit Departments

 2. The following modifications to sectoldif.cfg will cause an AIX 4.3 and AIX 5.1 compliant DIT and data to be created when sectoldif -d cn=aixsecdb,cn=aixdata -S aix is invoked:

     USER ou organizationalUnit aixuser
     GROUP ou organizationalUnit aixgroup
     ID ou organizationalUnit system

Related Information

The mksecldap command, the sectoldif command, and the nistoldif command.

Lightweight Directory Access Protocol in Operating system and device management.
security_default File

**Purpose**
Contains the NFS security defaults.

**Description**
The `/etc/nfs/security_default` file contains the list of security flavors that may be used by the NFS client, in the order in which they should be used. The list of valid security flavors are:

<table>
<thead>
<tr>
<th>Flavor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sys</td>
<td>Unix style (uids, gids)</td>
</tr>
<tr>
<td>dh</td>
<td>DES style (encrypted timestamps)</td>
</tr>
<tr>
<td>krb5</td>
<td>Kerberos 5, no integrity or privacy</td>
</tr>
<tr>
<td>krb5i</td>
<td>Kerberos 5, with integrity</td>
</tr>
<tr>
<td>krb5p</td>
<td>Kerberos 5, with privacy</td>
</tr>
</tbody>
</table>

This file may be modified using the `chnfssec` command.

The format of the `/etc/nfs/security_default` file is one security flavor (sys, krb5, etc.) per line.

**Files**
`/etc/nfs/security_default` The `security_default` file.

**Related Information**
The `chnfssec` command.

sendmail.cf File

**Purpose**
Contains the configuration information for the `sendmail` command.

**Description**
The `/etc/mail/sendmail.cf` configuration file contains the configuration information for the `sendmail` command. Information contained in this file includes such items as the host name and domain, and the `sendmail` rule sets.

The `/etc/mail/sendmail.cf` file:
- Stores information about the type of mailer programs running.
- Defines how the `sendmail` command rewrites addresses in messages.
- Defines how the `sendmail` command operates in the following environments:
  - Local mail delivery
  - Local area network delivery using TCP/IP
  - Remote delivery using Basic Utilities Network (BNU).

If your environment includes only these types of mail delivery, you can use the supplied `/etc/mail/sendmail.cf` file with few, if any, changes.

**Control Lines**
The `/etc/mail/sendmail.cf` file consists of a series of control lines, each of which begins with a single character defining how the rest of the line is used. Lines beginning with a space or a tab are continuation lines. Blank lines and lines beginning with a `#` (pound sign) are comments. Control lines are used for defining:
Rewrite Rules

The `sendmail` command receives addresses in a number of different formats because different mailers use different formats to deliver mail messages. The `sendmail` command changes the addresses to the format needed to route the message for the mailer program being used. To perform this translation, the `sendmail` command uses a set of rewrite rules, or *rule sets*, that are defined in the `/etc/mail/sendmail.cf` configuration file. Rewrite rules have the following format:

```
Snumber
Rbefore after
```

where `number` is an integer greater than or equal to zero indicating which rule set this is, and `before` and `after` are symbolic expressions representing a particular pattern of characters. The line beginning with `R` means rewrite the expression `before` so that it has the same format as the expression `after`. `Sendmail` scans through the set of rewrite rules looking for a match on the left-hand side (LHS) of the rule. When a rule matches, the address is replaced by the right-hand side (RHS) of the rule.

**Note:** There must be at least one TAB character (ASCII code 0x09) between the `before` and `after` sections of the `/etc/mail/sendmail.cf` file. For this reason, any editor that translates TAB characters into a series of spaces (ASCII code 0x20) may not be used to edit the `/etc/mail/sendmail.cf` file. For example, the GNU eMacs editor can corrupt the `sendmail.cf` file, but the vi editor does not.

The `/etc/mail/sendmail.cf` file installed with the `sendmail` command contains enough rules to perform the translation for BNU and TCP/IP networks using a domain address structure. You should not have to change these rules unless connecting to a system that uses a different addressing scheme.

Macro expansions of the form `$x` are performed when the configuration file is read. Expansions of the form `$&x` are performed at run time, using a somewhat less general algorithm. This form is intended only for referencing internally defined macros such as `$h` that are changed at runtime.

**Left-Hand Side (LHS) of Rewrite Rules:** The left-hand side of rewrite rules contains a pattern. Normal words are simply matched directly. Metasyntax is introduced using a dollar sign. The metasymbols are:

<table>
<thead>
<tr>
<th>Metasymbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$^*</td>
<td>Match zero or more tokens</td>
</tr>
<tr>
<td>$^+</td>
<td>Match one or more tokens</td>
</tr>
<tr>
<td>$^-</td>
<td>Match exactly one token</td>
</tr>
<tr>
<td>$^=x</td>
<td>Match any phrase in class <code>x</code></td>
</tr>
<tr>
<td>$^~x</td>
<td>Match any word not in class <code>x</code></td>
</tr>
</tbody>
</table>

If any of these match, they are assigned to the symbol `$n` for replacement on the right-hand side, where `n` is the index in the LHS. For example, if the LHS:

```
$^-:$+
```

is applied to the input:

```
UCBARPA:linda
```

the rule will match, and the values passed to the RHS will be:
**Right-Hand Side (RHS) of Rewrite Rules:** When the left-hand side of a rewrite rule matches, the input is deleted and replaced by the right-hand side. Tokens are copied directly from the RHS unless they begin with a dollar sign. Metasymbols are:

<table>
<thead>
<tr>
<th>Metasymbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n</td>
<td>Substitute indefinite token n from LHS</td>
</tr>
<tr>
<td>$[\text{name}]$</td>
<td>Canonicalize name</td>
</tr>
<tr>
<td>$(\text{map key}$@$\text{arguments}$:$\text{default}$)</td>
<td>Generalized keyed mapping function</td>
</tr>
<tr>
<td>$&gt;n</td>
<td>“Call” ruleset n</td>
</tr>
<tr>
<td>$@\text{host}$</td>
<td>Specify host</td>
</tr>
<tr>
<td>$\text{user}$</td>
<td>Specify user</td>
</tr>
</tbody>
</table>

The $n$ syntax substitutes the corresponding value from a $+$, $-$, $*$, $=$, or $~$ match on the LHS. It may be used anywhere.

A host name enclosed between $\{}$ and $\}$ is looked up in the host database(s) and replaced by the canonical name. For example, $\{\text{merlin}\}$ might become merlin.magician and $\{[[128.32.130.2]\}$ would become king.arthur.

The $(\ldots\})$ syntax is a more general form of lookup; it uses a named map instead of an implicit map. If no lookup is found, the indicated default is inserted; if no default is specified and no lookup matches, the value is left unchanged. The arguments are passed to the map for possible use.

The $>n$ syntax causes the remainder of the line to be substituted as usual and then passed as the argument to ruleset n. The final value of ruleset n then becomes the substitution for this rule. The $>n$ syntax can only be used at the beginning of the right hand side; it can only be preceded by $@$ or $:$.

The $#$ syntax should only be used in ruleset zero or a subroutine of ruleset zero. It causes evaluation of the ruleset to terminate immediately, and signals to sendmail that the address has completely resolved. The complete syntax is:

```
$\#\text{mailer}$@$\text{host}$:$\text{user}$
```

This specifies the (mailer, host, user) 3-tuple necessary to direct the mailer. If the mailer is local, the host part may be omitted. The mailer must be a single word, but the host and user may be multi-part. If the mailer is the built-in IPC mailer, the host may be a colon-separated list of hosts that are searched in order for the first working address, exactly like MX (machine exchange) records. The user is later rewritten by the mailer-specific envelope rewrite set and assigned to the $u$ macro. As a special case, if the value to $#$ is "local" and the first character of the $:$ value is "@", the "@" is stripped off, and a flag is set in the address descriptor that causes sendmail to not do ruleset 5 processing.

Normally, a rule that matches is retried, that is, the rule loops until it fails. An RHS may also be preceded by a $@$ or a $:$ to change this behavior. A $@$ prefix causes the ruleset to return with the remainder of the RHS as the value. A $:$ prefix causes the rule to terminate immediately, but the ruleset to continue; this can be used to avoid continued application of a rule. The prefix is stripped before continuing.

The $@$ and $:$ prefixes may precede a $>$ spec. For example:

```
R$+$ $: $>7 $1
```

matches anything, passes that to ruleset seven, and continues; the $:$ is necessary to avoid an infinite loop.
Substitution occurs in the order described; that is, parameters from the LHS are substituted, host names are canonicalized, "subroutines" are called, and finally $#, $@, and $: are processed.

**Semantics of Rewrite Rule Sets:** There are five rewrite sets that have specific semantics.

Ruleset three should turn the address into "canonical form." This form should have the basic syntax:

```
local-part@host-domain-spec
```

Ruleset three is applied by `sendmail` before doing anything with any address.

If no "@" sign is specified, then the host-domain-spec may be appended (box "D" in "Rewrite Set Semantics") from the sender address (if the C flag is set in the mailer definition corresponding to the sending mailer).

Ruleset zero is applied after ruleset three to addresses that are going to actually specify recipients. It must resolve to a {mailer, host, user} triple. The mailer must be defined in the mailer definitions from the configuration file. The host is defined into the $h macro for use in the argv expansion of the specified mailer.

**IPC Mailers:** Some special processing occurs if the ruleset zero resolves to an IPC mailer (that is, a mailer that has "[IPC]" listed as the Path in the M configuration line. The host name passed after "$@" has MX expansion performed; this looks the name up in DNS to find alternate delivery sites.

The host name can also be provided as a dotted quad in square brackets; for example:

```
[128.32.149.78]
```

This causes direct conversion of the numeric value to a TCP/IP host address.

The host name passed in after the "$@" may also be a colon-separated list of hosts. Each is separately MX expanded and the results are concatenated to make (essentially) one long MX list. The intent here is to create "fake" MX records that are not published in DNS for private internal networks.

As a final special case, the host name can be passed in as a text string in square brackets:

```
[any.internet.addr]
```

This form avoids the MX mapping if the F=0 flag is set for the selected delivery agent.

**Note:** This is intended only for situations where you have a network firewall (a system or machine that controls the access between outside networks and private networks) or other host that will do special processing for all your mail, so that your MX record points to a gateway machine. This machine could then do direct delivery to machines within your local domain. Use of this feature directly violates RFC 1123 section 5.3.5: it should not be used lightly.

**Macros in the sendmail.cf File**

Macros in the `/etc/mail/sendmail.cf` file are interpreted by the `sendmail` command. A macro is a symbol that represents a value or string. A macro is defined by a `D` command in the `/etc/mail/sendmail.cf` file.

**D — Define Macro:** Macros are named with a single character or with a word in {braces}. Single-character names may be selected from the entire ASCII set, but user-defined macros should be selected from the set of uppercase letters only. Lowercase letters and special symbols are used internally. Long names beginning with a lowercase letter or a punctuation character are reserved for use by `sendmail`, so user-defined long macro names should begin with an uppercase letter.

The syntax for macro definitions is:

```
Dxval
```
where \( x \) is the name of the macro (which may be a single character or a word in braces) and \( \text{val} \) is the value it should have. There should be no spaces given that do not actually belong in the macro value.

Macros are interpolated using the construct \( $x \), where \( x \) is the name of the macro to be interpolated. This interpolation is done when the configuration file is read, except in M lines. The special construct \( $&x \) can be used in R lines to get deferred interpolation.

Conditionals can be specified using the syntax:

\[
?x \text{ text1 } \mid \text{ text2 } \).
\]

This interpolates \( \text{text1} \) if the macro \( x \) is set, and \( \text{text2} \) otherwise. The "else" (\( \mid \)) clause may be omitted.

Lowercase macro names are reserved to have special semantics, used to pass information in or out of \texttt{sendmail}, and special characters are reserved to provide conditionals, and so on. Uppercase names (that is, \( $A \) through \( $Z \)) are specifically reserved for configuration file authors.

The following macros are defined and/or used internally by \texttt{sendmail} for interpolation into argv’s for mailers or for other contexts. The ones marked - are information passed into \texttt{sendmail}, the ones marked = are information passed both in and out of \texttt{sendmail}, and the unmarked macros are passed out of \texttt{sendmail} but are not otherwise used internally:

<table>
<thead>
<tr>
<th>Macro</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$$_$$</td>
<td>RFC1413-validation &amp; IP source route (V8.1 and above).</td>
</tr>
<tr>
<td>$a$$</td>
<td>The origin date in RFC822 format.</td>
</tr>
<tr>
<td>$b$$</td>
<td>The current date in RFC822 format.</td>
</tr>
<tr>
<td>$(bodytype)$$</td>
<td>The ESMTP BODY parameter.</td>
</tr>
<tr>
<td>$B$$</td>
<td>The BITNET relay.</td>
</tr>
<tr>
<td>$c$$</td>
<td>The hop count.</td>
</tr>
<tr>
<td>$client_addr$$</td>
<td>The connecting host’s IP address.</td>
</tr>
<tr>
<td>$client_name$$</td>
<td>The connecting host’s canonical name.</td>
</tr>
<tr>
<td>$client_port$$</td>
<td>The connecting host’s port name.</td>
</tr>
<tr>
<td>$client_resolve$$</td>
<td>Holds the result of the resolve call for $(client_name).</td>
</tr>
<tr>
<td>$(currHeader)$$</td>
<td>Header value as quoted string</td>
</tr>
<tr>
<td>$C$$</td>
<td>The hostname of the DECnet relay (m4 technique).</td>
</tr>
<tr>
<td>$d$$</td>
<td>The current date in UNIX (\texttt{ctime}) format.</td>
</tr>
<tr>
<td>$(daemon_addr)$$</td>
<td>The IP address on which the daemon is listening for connections. Unless \texttt{DaemonPortOptions} is set, this will be 0.0.0.0.</td>
</tr>
<tr>
<td>$(daemon_family)$$</td>
<td>If the daemon is accepting network connections, this is the network family.</td>
</tr>
<tr>
<td>$(daemon_flags)$$</td>
<td>The flags for the daemon as specified by the Modifiers= part of \texttt{DaemonPortOptions} where the flags are separated from each other by spaces and upper case flags are doubled.</td>
</tr>
<tr>
<td>$(daemon_info)$$</td>
<td>Information about a daemon as a text string. For example, \texttt{SMTP+queueing@00}.</td>
</tr>
<tr>
<td>$(daemon_name)$$</td>
<td>The name of the daemon from \texttt{DaemonPortOptions} Name= suboption. If this suboption is not used, the default will be set to Daemon#, where # is the daemon number.</td>
</tr>
<tr>
<td>$(daemon_port)$$</td>
<td>The port on which the daemon is accepting connections. Unless \texttt{DaemonPortOptions} is set, this will most likely be set to the default of 25.</td>
</tr>
<tr>
<td>$(deliveryMode)$$</td>
<td>The current delivery mode used by \texttt{sendmail}.</td>
</tr>
<tr>
<td>$e$$</td>
<td>Obsolete. Used \texttt{SmtpGreetingMessage} option instead.</td>
</tr>
<tr>
<td>$envid$$</td>
<td>The original DSN envelope ID.</td>
</tr>
<tr>
<td>$E$$</td>
<td>X400 relay (unused) (m4 technique).</td>
</tr>
<tr>
<td>$f$$</td>
<td>The sender’s address.</td>
</tr>
<tr>
<td>$F$$</td>
<td>FAX relay (m4 technique).</td>
</tr>
<tr>
<td>$g$$</td>
<td>The sender’s address relative to the recipient.</td>
</tr>
<tr>
<td>$h$$</td>
<td>Host part of the recipient address.</td>
</tr>
<tr>
<td>$H$$</td>
<td>The mail hub (m4 technique).</td>
</tr>
<tr>
<td>$hdrlen$$</td>
<td>The length of the header value, which is stored in $(currHeader).</td>
</tr>
<tr>
<td>Macro</td>
<td>Definition</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>$(hdr_name)</td>
<td>The name of the header field for which the current header check ruleset has been called.</td>
</tr>
<tr>
<td>$i</td>
<td>The queue identifier.</td>
</tr>
<tr>
<td>$i(addr)</td>
<td>The IP address of an incoming connection interface unless it is in the loopback net.</td>
</tr>
<tr>
<td>$i(name)</td>
<td>The name of an incoming connection interface.</td>
</tr>
<tr>
<td>$i=</td>
<td>The official canonical name.</td>
</tr>
<tr>
<td>$k</td>
<td>The UUCP node name (V8.1 and above).</td>
</tr>
<tr>
<td>$l</td>
<td>Obsolete. Use <strong>UnixFromLine</strong> option instead.</td>
</tr>
<tr>
<td>$L</td>
<td>Local user relay (m4 technique).</td>
</tr>
<tr>
<td>$m</td>
<td>The DNS domain name (V8.1 and above).</td>
</tr>
<tr>
<td>$M</td>
<td>Who we are masquerading as (m4 technique).</td>
</tr>
<tr>
<td>$(mail_addr)</td>
<td>The address part of the resolved triple of the address given for the SMTP MAIL command.</td>
</tr>
<tr>
<td>$(mail_host)</td>
<td>The host from the resolved triple of the address given for the SMTP MAIL command.</td>
</tr>
<tr>
<td>$(mail_mailer)</td>
<td>The mailer from the resolved triple of the address given for the SMTP MAIL command.</td>
</tr>
<tr>
<td>$n</td>
<td>The error messages sender.</td>
</tr>
<tr>
<td>$n(tries)</td>
<td>The number of delivery attempts.</td>
</tr>
<tr>
<td>$o</td>
<td>Obsolete. Use <strong>OperatorChars</strong> option instead.</td>
</tr>
<tr>
<td>$opMode</td>
<td>The startup operating mode (V8.7 and above).</td>
</tr>
<tr>
<td>$p</td>
<td>The <strong>sendmail</strong> process ID.</td>
</tr>
<tr>
<td>$(queue_interval)</td>
<td>The queue run interval as defined in the -q flag.</td>
</tr>
<tr>
<td>$r</td>
<td>The protocol used.</td>
</tr>
<tr>
<td>$R</td>
<td>The relay for unqualified names (m4 technique).</td>
</tr>
<tr>
<td>$(rcpt_addr)</td>
<td>The address part of the resolved triple of the address given for the SMTP RCPT command.</td>
</tr>
<tr>
<td>$(rcpt_host)</td>
<td>The host from the resolved triple of the address given for the SMTP RCPT command.</td>
</tr>
<tr>
<td>$(rcpt_mailer)</td>
<td>The mailer from the resolved triple of the address given for the SMTP RCPT command.</td>
</tr>
<tr>
<td>$s</td>
<td>The sender's host name.</td>
</tr>
<tr>
<td>$S</td>
<td>The Smart host (m4 technique).</td>
</tr>
<tr>
<td>$(server_addr)</td>
<td>The address of the server of the current outgoing SMTP connection.</td>
</tr>
<tr>
<td>$(server_name)</td>
<td>The name of the server of the current outgoing SMTP connection.</td>
</tr>
<tr>
<td>$t</td>
<td>Current time in seconds.</td>
</tr>
<tr>
<td>$u</td>
<td>The recipient's user name.</td>
</tr>
<tr>
<td>$U</td>
<td>The UUCP name to override $k.</td>
</tr>
<tr>
<td>$v</td>
<td>The <strong>sendmail</strong> program's version.</td>
</tr>
<tr>
<td>$V</td>
<td>The UUCP relay (for class $=V) (m4 technique).</td>
</tr>
<tr>
<td>$w</td>
<td>The short name of this host.</td>
</tr>
<tr>
<td>$W</td>
<td>The UUCP relay (for class $=W) (m4 technique).</td>
</tr>
<tr>
<td>$x</td>
<td>The full name of the sender.</td>
</tr>
<tr>
<td>$X</td>
<td>The UUCP relay (for class $=X) (m4 technique).</td>
</tr>
<tr>
<td>$y</td>
<td>The home directory of the recipient.</td>
</tr>
<tr>
<td>$Y</td>
<td>The UUCP relay for unclassified hosts.</td>
</tr>
<tr>
<td>$z</td>
<td>The recipient's home directory.</td>
</tr>
<tr>
<td>$Z</td>
<td>The version of this m4 configuration (m4 technique).</td>
</tr>
</tbody>
</table>

There are three types of dates that can be used. The $a$ and $b$ macros are in RFC 822 format; $a$ is the time as extracted from the “Date:” line of the message (if there was one), and $b$ is the current date and time (used for postmarks). If no “Date:” line is found in the incoming message, $a$ is set to the current time also. The $d$ macro is equivalent to the $b$ macro in UNIX (ctime) format. The $t$ macro is the current time in seconds.
The macros $w$, $j$, and $m$ are set to the identity of this host. Sendmail tries to find the fully qualified name of the host if at all possible; it does this by calling gethostbyname(2) to get the current hostname and then passing that to gethostbyname(3) which is supposed to return the canonical version of that host name. Assuming this is successful, $j$ is set to the fully qualified name, and $m$ is set to the domain part of the name (everything after the first dot). The $w$ macro is set to the first word (everything before the first dot) if you have a level 5 or higher configuration file; otherwise, it is set to the same value as $j$. If the canonicalization is not successful, it is imperative that the config file set $j$ to the fully qualified domain name.

The $f$ macro is the ID of the sender as originally determined; when mailing to a specific host, the $g$ macro is set to the address of the sender relative to the recipient. For example, if a user sends to king@castle.com from the machine vangogh.painter.com, the $f$ macro will be vincent and the $g$ macro will be vincent@vangogh.painter.com.

The $x$ macro is set to the full name of the sender. This can be determined in several ways. It can be passed as flag to sendmail. It can be defined in the NAME environment variable. The third choice is the value of the "Full-Name:" line in the header if it exists, and the fourth choice is the comment field of a "From:" line. If all of these fail, and if the message is being originated locally, the full name is looked up in the /etc/passwd file.

When sending, the $h$, $u$, and $z$ macros get set to the host, user, and home directory (if local) of the recipient. The first two are set from the $@$ and $:$ part of the rewrite rules, respectively.

The $p$ and $t$ macros are used to create unique strings (for example, for the "Message-Id:" field). The $i$ macro is set to the queue ID on this host; if put into the timestamp line, it can be useful for tracking messages. The $v$ macro is set to the version number of sendmail; this is normally put in timestamps and has been proven useful for debugging.

The $c$ field is set to the "hop count," that is, the number of times this message has been processed. This can be determined by the -h flag on the command line or by counting the timestamps in the message.

The $r$ and $s$ fields are set to the protocol used to communicate with sendmail and the sending hostname. They can be set together using the -p command line flag or separately using the -M or -oM flags.

The $_$ is set to a validated sender host name. If the sender is running an RFC 1413 compliant IDENT server and the receiver has the IDENT protocol turned on, it will include the user name on that host.

The $(client_name), $(client_addr), and $(client_port) macros are set to the name, address, and port number of the connecting host who is invoking sendmail as a server. These can be used in the check_* rulesets (using the $& deferred evaluation form).

**Changing the Domain Name Macro:**

**Note:** This function is available in AIX 4.1 only.

The domain name macro, DD, specifies the full domain name of your local group of hosts. The format of the domain name macro is DD followed by, at most, four period-separated names, for example:

```
DDname1.name2.name3.name4
```

This macro can be set automatically through the `hostname` command. The `sendmail` command reads what has been set with the `hostname` command and uses it to initialize the host and domain macros and classes. The configuration file macros only need to be changed if you want the `sendmail` host and domain names to be different from those set by the `hostname` command.

To change the domain name macro:
1. Enter the command:
   ```bash
   vi /etc/mail/sendmail.cf
   ```

2. Find the line beginning with `DD`.

3. Replace what follows `DD` with your domain name. For example, if your domain name is `newyork.abc.com`, enter:
   ```plaintext
   DDnewyork.abc.com
   ```

4. Save the file and exit the editor.

**Changing the Host Name Macro:** The host name macro, `Dw`, specifies the name of your host system used in the return address of all messages you generate. The format of the host name macro is `Dw` followed by the hostname of this machine, for example:

```plaintext
Dwhostname
```

By default, the `sendmail` command reads what has been set with the `hostname` command and uses it to initialize the host and domain name macros and classes. Change the configuration file macros only if you want the `sendmail` command host and domain names to be different from those set by the `hostname` command.

To change the host name macro:

1. Enter the command:
   ```bash
   vi /etc/mail/sendmail.cf
   ```

2. Find the line beginning with `Dw`.

3. Replace what follows `Dw` with your hostname. For example, if your hostname is `brown`, enter:
   ```plaintext
   Dwbrown
   ```

4. Save the file and exit the editor.

   **Note:** If the `Dw` macro is defined, you must also define the `CW` (hostname) class.

**Modifying the sendmail.cf File**

Before you modify the `/etc/mail/sendmail.cf` file, make a backup copy. Do this by executing the following command:

```bash
cp /etc/mail/sendmail.cf /etc/mail/sendmail.cf.working
```

If the changes you make cause the mail system not to work properly, you can return to using a copy of the `/etc/mail/sendmail.cf` file that you know works.

You can modify the `/etc/mail/sendmail.cf` file by using your favorite text editor. However, some editors store tabs as the number of spaces they represent, not the tab character itself. This can cause unexpected results if the tab character is defined as the field-separator character in rule sets. Use the vi editor to avoid this problem, or change the field-separator character with the `J` option. (For ease of reference, this discussion assumes you use the vi editor to modify the `/etc/mail/sendmail.cf` file.)

After changing any information in the `/etc/mail/sendmail.cf` file, you must instruct the daemon to reread the file. See section, “Making the sendmail Daemon Reread the Configuration Information” for those instructions.

**Making the sendmail Daemon Reread the Configuration Information:** After you have made changes to the `sendmail.cf` file, instruct the daemon to reread the file. If you started the `sendmail` command using the `startsrc` command, enter the command:

```bash
refresh -s sendmail
```

Or, if you started the `sendmail` daemon using the `/usr/sbin/sendmail` command, enter the command:

```bash
kill -l `cat /etc/mail/sendmail.pid`
```
Both of these commands cause the daemon to reread the `/etc/mail/sendmail.cf` file, the `/etc/mail/aliases` file, and the `/etc/sendmail.nl` file.

**Alias Database**

The alias database exists in two forms. One is a text form, maintained in the file `/etc/mail/aliases`. The aliases are of the form:

name: name1, name2, ...

Only local names may be aliased. For example:

linda@cloud.ai.acme.org: linda@CS.

has the desired effect. Aliases may be continued by starting any continuation lines with a space or a tab. Blank lines and lines beginning with a pound sign (#) are comments.

The second form is processed by the new database manager (NDBM) or Berkeley DB library. This form is in the file `/etc/mail/aliases.db` (if using NEWDB) or `/etc/mail/aliases.dir` and `/etc/mail/aliases.pag` (if using NDBM). This is the form that `sendmail` actually uses to resolve aliases. This technique is used to improve performance.

The service switch sets the control of search order. The following entry

`AliasFile=switch:aliases`

is always added as the first alias entry. The first alias file name without a class (for example, without `nis` on the front) will be used as the name of the file for a "files" entry in the aliases switch. For example, if the configuration file contains

`AliasFile=/etc/mail/aliases`

and the service switch contains

`aliases nis files nisplus`

then aliases will first be searched in the NIS database, then in `/etc/mail/aliases`, and finally in the NIS+ database.

**Rebuilding the Alias Database:** The DB or DBM version of the database may be rebuilt explicitly by executing the command:

`newaliases`

This is equivalent to giving `sendmail` the `-bi` flag:

`/usr/sbin/sendmail -bi`

If the `RebuildAliases` option is specified in the configuration, `sendmail` will rebuild the alias database automatically if possible when it is out of date. Auto-rebuild can be dangerous on heavily loaded machines with large alias files. If it might take more than the rebuild time-out (option `AliasWait`, which is normally five minutes) to rebuild the database, there is a chance that several processes will start the rebuild process simultaneously.

If you have multiple aliases databases specified, the `-bi` flag rebuilds all the database types. It understands, for example, it can rebuild NDBM databases, but not NIS databases.

**Potential Problems with the Alias Database:** There are a number of problems that can occur with the alias database. They all result from a `sendmail` process accessing the DBM version while it is only partially built. This can happen under two circumstances: One process accesses the database while another process is rebuilding it, or the process rebuilding the database dies (due to being killed or a system crash) before completing the rebuild.
Sendmail has three techniques to try to relieve these problems. First, it ignores interrupts while rebuilding the database; this avoids the problem of someone aborting the process leaving a partially rebuilt database. Second, it locks the database source file during the rebuild, but that may not work over NFS or if the file is not writable. Third, at the end of the rebuild, it adds an alias of the form:

@: @

(which is not normally legal). Before sendmail will access the database, it checks to ensure that this entry exists.

List Owners: If an error occurs on sending to a certain address, x, sendmail will look for an alias of the form owner-x to receive the errors. This is typically useful for a mailing list where the submitter of the list has no control over the maintenance of the list itself. In this case, the list maintainer would be the owner of the list. For example:

unix-wizards: linda@paintbox, wnj@monet, nosuchuser, sam@matisse
owner-unix-wizards: unix-wizards-request
unix-wizards-request: linda@paintbox

would cause linda@paintbox to get the error that will occur when someone sends to unix-wizards due to the inclusion of nosuchuser on the list.

List owners also cause the envelope sender address to be modified. The contents of the owner alias are used if they point to a single user. Otherwise, the name of the alias itself is used. For this reason, and to conform to Internet conventions, the “owner-” address normally points at the “-request” address; this causes messages to go out with the typical Internet convention of using “list-request” as the return address.

Per-User Forwarding (.forward Files)
As an alternative to the alias database, users may put a file with the name “.forward” in their home directory. If this file exists, sendmail redirects mail for that user to the list of addresses listed in the .forward file. For example, if the home directory for user “kenly” has a .forward file with contents:

kenly@ernie
joel@re Noir

then any mail arriving for “kenly” will be redirected to the specified accounts.

The configuration file defines a sequence of file names to check. By default, this is the user’s .forward file, but can be defined to be more general using the ForwardPath (J) option. If you change this option, you must inform your user base of the change.

IDENT Protocol Support
UCB sendmail supports the IDENT protocol as defined in RFC 1413. Although this enhances identification of the author of an e-mail message by doing a “callback” to the originating system to include the owner of a particular TCP connection in the audit trail, it is in no sense perfect; a determined forger can easily violate the security of the IDENT protocol.

Note: The operating system does not support the IDENT protocol. The IDENT time-out is set to zero (0) in the /etc/mail/sendmail.cf file to disable IDENT. Modify your sendmail.cf file and set IDENT time-out if you wish to enable IDENT.

The following description is excerpted from RFC 1413:

6. Security Considerations

The information returned by this protocol is at most as trustworthy as the host providing it OR the organization operating the host. For example, a PC in an open lab has few if any controls on it to prevent
a user from having this protocol return any identifier the user wants. Likewise, if the host has been compromised the information returned may be completely erroneous and misleading.

The Identification Protocol is not intended as an authorization or access control protocol. At best, it provides some additional auditing information with respect to TCP connections. At worst, it can provide misleading, incorrect, or maliciously incorrect information.

The use of the information returned by this protocol for other than auditing is strongly discouraged. Specifically, using Identification Protocol information to make access control decisions, either as the primary method (that is, no other checks) or as an adjunct to other methods may result in a weakening of normal host security.

An Identification server may reveal information about users, entities, objects or processes which might normally be considered private. An Identification server provides service which is a rough analog of the CallerID services provided by some phone companies and many of the same privacy considerations and arguments that apply to the CallerID service apply to Identification. If you would not run a “finger” server due to privacy considerations you may not want to run this protocol.

**Tuning**

There are a number of configuration parameters you may want to change, depending on the requirements of your site. Most of these are set using an option in sendmail.cf. For example, the line “O Time-out.queuereturn=5d” sets option “Timeout.queuereturn” to the value “5d” (five days).

Most of these options have appropriate defaults for most sites. However, sites having very high mail loads may find they need to tune them as appropriate for their mail load. In particular, sites experiencing a large number of small messages, many of which are delivered to many recipients, may find that they need to adjust the parameters dealing with queue priorities.

All prior versions of sendmail had single-character option names. Although old short names are still accepted, most new options do not have short equivalents.

**Timeouts:** All time intervals are set using a scaled syntax. For example, “10m” represents ten minutes, whereas “2h30m” represents two and a half hours. The full set of scales is:

- **s** seconds
- **m** minutes
- **h** hours
- **d** days
- **w** weeks

**Read Timeouts:** Timeouts all have option names ”Time-out.suboption”. The recognized suboptions, their default values, and the minimum values allowed by RFC 1123 section 5.3.2 are:

<table>
<thead>
<tr>
<th>Suboption</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>command-</td>
<td>In server SMTP, the time to wait for another command. [1h, 5m].</td>
</tr>
<tr>
<td>connect</td>
<td>The time to wait for an SMTP connection to open (the connect(2) system call) [0, unspecified]. If zero, uses the kernel default. In no case can this option extend the time-out longer than the kernel provides, but it can shorten it. This is to get around kernels that provide an extremely long connection time-out (90 minutes in one case).</td>
</tr>
<tr>
<td>control</td>
<td>The time-out for a complete control socket transaction to complete [2m, none].</td>
</tr>
<tr>
<td>datablock-</td>
<td>The wait for reading a data block (that is, the body of the message). [1h, 3m]. This should be long because it also applies to programs piping input to sendmail which have no guarantee of promptness.</td>
</tr>
<tr>
<td>Suboption</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
</tr>
<tr>
<td>datafinal-</td>
<td>The wait for a reply from the dot terminating a message. [1h, 10m]. If this is shorter than the time actually needed for the receiver to deliver the message, duplicates will be generated. This is discussed in RFC1047.</td>
</tr>
<tr>
<td>datainit-</td>
<td>The wait for a reply from a DATA command [5m, 2m].</td>
</tr>
<tr>
<td>fileopen</td>
<td>The time-out for opening .forward and :include:.files [60s, none].</td>
</tr>
<tr>
<td>iconnect</td>
<td>The same as connect, except it applies only to the initial attempt to connect to a host for a given message [0, unspecified]. This period should be very short (a few seconds). Hosts that are well-connected and responsive will be serviced immediately. Hosts that are slow do not detain other deliveries in the initial delivery attempt.</td>
</tr>
<tr>
<td>ident-</td>
<td>The time-out waiting for a reply to an IDENT query [30s11, unspecified].</td>
</tr>
<tr>
<td>initial</td>
<td>The wait for the initial 220 greeting message [5m, 5m].</td>
</tr>
<tr>
<td>helo</td>
<td>The wait for a reply from a HELO or EHLO command [5m, unspecified]. This may require a host name lookup, so five minutes is probably a reasonable minimum.</td>
</tr>
<tr>
<td>hoststatus</td>
<td>The time that long status information about a host (for example, host down) will be cached before it is considered stale [30m, unspecified].</td>
</tr>
<tr>
<td>mail-</td>
<td>The wait for a reply from a MAIL command [10m, 5m].</td>
</tr>
<tr>
<td>misc</td>
<td>The wait for a reply from miscellaneous (but short) commands such as NOOP (no-operation) and VERB (go into verbose mode). [2m, unspecified].</td>
</tr>
<tr>
<td>quit</td>
<td>The wait for a reply from a QUIT command [2m, unspecified].</td>
</tr>
<tr>
<td>rcpt-</td>
<td>The wait for a reply from a RCPT command [1h, 5m]. This should be long because it could be pointing at a list that takes a long time to expand (see below).</td>
</tr>
<tr>
<td>rset</td>
<td>The wait for a reply from a RSET command [5m, unspecified].</td>
</tr>
<tr>
<td>resolver.retrans</td>
<td>Sets resolver retransmission time interval in seconds. Sets both the Timeout.resolver.retrans.first and Timeout.resolver.retrans.normal.</td>
</tr>
<tr>
<td>resolver.retrans.first</td>
<td>Sets resolver retransmission time interval in seconds for the first attempt to deliver a message.</td>
</tr>
<tr>
<td>resolver.retrans.normal</td>
<td>Sets the retransmission time interval in seconds for all resolver lookups except for the first delivery attempt.</td>
</tr>
<tr>
<td>resolver.retry</td>
<td>Sets the number of attempts to retransmit a resolver query. Sets both Timeout.resolver.retry.first and Timeout.resolver.retry.normal.</td>
</tr>
<tr>
<td>resolver.retry.first</td>
<td>Sets the number of attempts to retransmit a resolver query for the first delivery attempt.</td>
</tr>
<tr>
<td>resolver.retry.normal</td>
<td>Sets the number of attempts to retransmit a resolver query for all resolver lookups except the first delivery attempt.</td>
</tr>
</tbody>
</table>

For compatibility with old configuration files, if no suboption is specified, all the timeouts marked with - are set to the indicated value.

**Message Timeouts:** After sitting in the queue for a few days, a message will time out. This is to ensure that at least the sender is aware of the inability to send a message. The time-out is typically set to five days. It is sometimes considered convenient to also send a warning message if the message is in the queue longer than a few hours (assuming you normally have good connectivity; if your messages normally took several hours to send, you would not want to do this because it would not be an unusual event). These timeouts are set using the Timeout.queuereturn and Timeout.queuewarn options in the configuration file (previously both were set using the T option).

Because these options are global and you cannot know how long another host outside your domain will be down, a five-day time-out is recommended. This allows a recipient to fix the problem even if it occurs at the beginning of a long weekend. RFC 1123 section 5.3.1.1 says that this parameter should be “at least 4-5 days”.

The Timeout.queuewarn value can be piggybacked on the T option by indicating a time after which a warning message should be sent; the two timeouts are separated by a slash. For example, the line:

OT5d/4h
causes e-mail to fail after five days, but a warning message will be sent after four hours. This should be large enough that the message will have been tried several times.

Queue interval: The argument to the -q flag specifies how often a subdaemon will run the queue. This is typically set to between fifteen minutes and one hour. RFC 1123, section 5.3.1.1 recommends this be at least 30 minutes.

Forking During Queue Runs: By setting the ForkEachJob (Y) option, sendmail will fork before each individual message while running the queue. This will prevent sendmail from consuming large amounts of memory, so it may be useful in memory-poor environments. However, if the ForkEachJob option is not set, sendmail will keep track of hosts that are down during a queue run, which can improve performance dramatically.

If the ForkEachJob option is set, sendmail cannot use connection caching.

Queue Priorities: Every message is assigned a priority when it is first instantiated, consisting of the message size (in bytes) offset by the message class (which is determined from the Precedence: header) times the “work class factor” and the number of recipients times the “work recipient factor.” The priority is used to order the queue. Higher numbers for the priority mean that the message will be processed later when running the queue.

The message size is included so that large messages are penalized relative to small messages. The message class allows users to send “high priority” messages by including a “Precedence:” field in their message; the value of this field is looked up in the P lines of the configuration file. Because the number of recipients affects the amount of load a message presents to the system, this is also included into the priority.

The recipient and class factors can be set in the configuration file using the RecipientFactor (y) and ClassFactor (z) options respectively. They default to 30000 (for the recipient factor) and 1800 (for the class factor). The initial priority is:

\[
\text{pri} = \text{msgsize} - (\text{class} \times \text{ClassFactor}) + (\text{nrcpt} \times \text{RecipientFactor})
\]

(Remember that higher values for this parameter actually mean that the job will be treated with lower priority.)

The priority of a job can also be adjusted each time it is processed (that is, each time an attempt is made to deliver it) using the “work time factor,” set by the RetryFactor(Z) option. This is added to the priority, so it normally decreases the precedence of the job, on the grounds that jobs that have failed many times will tend to fail again in the future. The RetryFactor option defaults to 90000.

Load Limiting: Sendmail can be asked to queue (but not deliver) mail if the system load average gets too high using the QueueLA (x) option. When the load average exceeds the value of the QueueLA option, the delivery mode is set to q (queue only) if the QueueFactor (q) option divided by the difference in the current load average and the QueueLA option plus one exceeds the priority of the message; that is, the message is queued if:

\[
\text{pri} > \left\{ \text{bold QueueFactor} \right\} \text{over} \left\{ \text{LA} - \left\{ \text{bold QueueLA} \right\} + 1 \right\}
\]

The QueueFactor option defaults to 600000, so each point of load average is worth 600000 priority points (as described above).

For drastic cases, the RefuseLA (X) option defines a load average at which sendmail will refuse to accept network connections. Locally generated mail (including incoming UUCP mail) is still accepted.

Delivery Mode: There are a number of delivery modes that sendmail can operate in, set by the DeliveryMode (d) configuration option. These modes specify how quickly mail will be delivered. Legal
modes are:

<table>
<thead>
<tr>
<th>Delivery Mode</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>Deliver interactively (synchronously)</td>
</tr>
<tr>
<td>b</td>
<td>Deliver in background (asynchronously)</td>
</tr>
<tr>
<td>q</td>
<td>Queue only (do not deliver)</td>
</tr>
<tr>
<td>d</td>
<td>Defer delivery attempts (do not deliver)</td>
</tr>
</tbody>
</table>

There are trade-offs. Mode i gives the sender the quickest feedback, but may slow down some mailers and is hardly ever necessary. Mode b delivers promptly, but can cause large numbers of processes if you have a mailer that takes a long time to deliver a message. Mode q minimizes the load on your machine, but means that delivery may be delayed for up to the queue interval. Mode d is identical to mode q except that it also prevents all the early map lookups from working; it is intended for “dial on demand” sites where DNS lookups might be very expensive. Some simple error messages (for example, host unknown during the SMTP protocol) will be delayed using this mode. Mode b is the default.

If you run in mode q (queue only), d (defer), or b (deliver in background), sendmail will not expand aliases and follow .forward files upon initial receipt of the mail. This speeds up the response to RCPT commands. Mode i cannot be used by the SMTP server.

**Log Level:** The level of logging can be set for sendmail. The default using a standard configuration table is level 9. The levels are as follows:

<table>
<thead>
<tr>
<th>Log Level</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Minimum logging.</td>
</tr>
<tr>
<td>1</td>
<td>Serious system failures and potential security problems.</td>
</tr>
<tr>
<td>2</td>
<td>Lost communications (network problems) and protocol failures.</td>
</tr>
<tr>
<td>3</td>
<td>Other serious failures.</td>
</tr>
<tr>
<td>4</td>
<td>Minor failures.</td>
</tr>
<tr>
<td>5</td>
<td>Message collection statistics.</td>
</tr>
<tr>
<td>6</td>
<td>Creation of error messages, VRFY and EXPN commands.</td>
</tr>
<tr>
<td>7</td>
<td>Delivery failures (for example, host or user unknown).</td>
</tr>
<tr>
<td>8</td>
<td>Successful deliveries and alias database rebuilds.</td>
</tr>
<tr>
<td>9</td>
<td>Messages being deferred (for example, due to a host being down).</td>
</tr>
<tr>
<td>10</td>
<td>Database expansion (alias, forward, and userdb lookups).</td>
</tr>
<tr>
<td>11</td>
<td>NIS errors and end-of-job processing.</td>
</tr>
<tr>
<td>12</td>
<td>Logs all SMTP connections.</td>
</tr>
<tr>
<td>13</td>
<td>Logs bad user shells, files with improper permissions, and other questionable situations.</td>
</tr>
<tr>
<td>14</td>
<td>Logs refused connections.</td>
</tr>
<tr>
<td>15</td>
<td>Log all incoming and outgoing SMTP commands.</td>
</tr>
<tr>
<td>20</td>
<td>Logs attempts to run locked queue files. These are not errors, but can be useful to note if your queue appears to be clogged.</td>
</tr>
<tr>
<td>30</td>
<td>Lost locks (only if using lockf instead of flock).</td>
</tr>
</tbody>
</table>

**File Modes:** The modes used for files depend on what functionality you want and the level of security you require.

The database that sendmail actually uses is represented by the following file:

/etc/mail/aliases.db

Berkeley DB database

The mode on these files should match the mode of /etc/mail/aliases.db. If aliases is writable and the files are not, users will be unable to reflect their desired changes through to the actual database. However, if aliases is read-only and DBM files are writable, a slightly sophisticated user can arrange to steal mail anyway.
If your DBM files are not writable, or you do not have auto-rebuild enabled (with the AutoRebuildAliases option), then you must be careful to reconstruct the alias database each time you change the text version:

```
newaliases
```

If this step is ignored or forgotten, any intended changes will be lost.

**Connection Caching:** When processing the queue, `sendmail` will try to keep the last few open connections open to avoid startup and shutdown costs. This only applies to IPC connections.

When trying to open a connection, the cache is first searched. If an open connection is found, it is probed to see if it is still active by sending a `RSET` command. It is not an error if this fails; instead, the connection is closed and reopened.

Two parameters control the connection cache. The `ConnectionCacheSize` (k) option defines the number of simultaneous open connections that will be permitted. If it is set to zero, connections will be closed as quickly as possible. The default is one. This should be set as appropriate for your system size; it will limit the amount of system resources that `sendmail` will use during queue runs. Never set this higher than 4.

The `ConnectionCacheTimeout` (K) option specifies the maximum time that any cached connection will be permitted to idle. When the idle time exceeds this value, the connection is closed. This number should be small (under ten minutes) to prevent you from grabbing too many resources from other hosts. The default is five minutes.

**Name Server Access:** If you want machine exchange (MX) support, you must be using Domain Name Services (DNS).

The `ResolverOptions(I)` option allows you to tweak name server options. The command line takes a series of flags as documented in `resolver(3)` (with the leading "RES_" deleted). Each can be preceded by an optional `+` or `-`. For example, the line:

```
O ResolverOptions=+AAONLY -DNSRCH
```

turns on the AAONLY (Accept Authoritative Answers only) and turns off the DNSRCH (search the domain path) options. Most resolver libraries default DNSRCH, DEFNAMES, and RECURSE flags on and all others off. You can also include "HasWildcardMX" to specify that there is a wildcard MX record matching your domain; this turns off MX matching when canonicalizing names, which can lead to inappropriate canonicalizations.

**Moving the Per-User Forward Files:** Some sites mount each user's home directory from a local disk on their workstation, so that local access is fast. However, the result is that `.forward` file lookups are slow. In some cases, mail can even be delivered on machines inappropriately because of a file server being down. The performance can be especially bad if you run the automounter.

The `ForwardPath (J)` option allows you to set a path of forward files. For example, the config file line:

```
O ForwardPath=/var/forward/\$u:\$z/.forward.$w
```

would first look for a file with the same name as the user's login in `/var/forward`. If that is not found (or is inaccessible), the file ".forward.machinename" in the user's home directory is searched.

If you create a directory such as `/var/forward`, it should be mode 1777 (that is, the sticky bit should be set). Users should create the files mode 644.

**Free Space:** On systems that have one of the system calls in the `statfs(2)` family (including `statvfs` and `ustat`), you can specify a minimum number of free blocks on the queue file system using the `MinFreeBlocks (b)` option. If there are fewer than the indicated number of blocks free on the filesystem on which the queue is mounted, the SMTP server will reject mail with the 452 error code. This invites the SMTP client to try again later.
**Attention:** Be careful not to set this option too high; it can cause rejection of e-mail when that mail would be processed without difficulty.

**Maximum Message Size:** To avoid overflowing your system with a large message, the MaxMessageSize option can set an absolute limit on the size of any one message. This will be advertised in the ESMTP dialogue and checked during message collection.

**Privacy Flags:** The PrivacyOptions (p) option allows you to set certain "privacy" flags. Actually, many of them do not give you any extra privacy, rather just insisting that client SMTP servers use the HELO command before using certain commands or adding extra headers to indicate possible security violations.

The option takes a series of flag names; the final privacy is the inclusive or of those flags. For example:

```
0 PrivacyOptions=needmailhelo, noexpn
```

insists that the HELO or EHLO command be used before a MAIL command is accepted and disables the EXPN command.

The flags are detailed in RFC 1123 S 5.1.6.

**Send to Me Too:** Normally, sendmail deletes the (envelope) sender from any list expansions. For example, if "linda" sends to a list that contains "linda" as one of the members, she will not get a copy of the message. If the -m (me too) command line flag, or if the MeToo (m) option is set in the configuration file, this behavior is suppressed.

**C and F — Define Classes:** Classes of phrases may be defined to match on the left hand side of rewrite rules, where a "phrase" is a sequence of characters that do not contain space characters. For example, a class of all local names for this site might be created so that attempts to send to oneself can be eliminated. These can either be defined directly in the configuration file or read in from another file. Classes are named as a single letter or a word in {braces}. Class names beginning with lowercase letters and special characters are reserved for system use. Classes defined in config files may be given names from the set of uppercase letters for short names or beginning with an uppercase letter for long names.

```
Ccphrase1 phrase2...
Fcfile
```

The first form defines the class c to match any of the named words. It is permissible to split them among multiple lines; for example, the two forms:

```
CHmonet ucbmonet
```

and

```
CHmonet
CHucbmonet
```

are equivalent. The "F" form reads the elements of the class c from the named file.

Elements of classes can be accessed in rules using $= or $~. The $~ (match entries not in class) only matches a single word; multi-word entries in the class are ignored in this context.

The class $=w is set to be the set of all names this host is known by. This can be used to match local host names.

The class $=k is set to be the same as $k, that is, the UUCP node name.

The class $=m is set to the set of domains by which this host is known, initially just $m.

The class $=t is set to the set of trusted users by the T configuration line. If you want to read trusted users from a file, use **Ft/file/name**.
The class $=n can be set to the set of MIME body types that can never be eight to seven bit encoded. It defaults to "multipartSigned". Message types "message/*" and "multipart/*" are never encoded directly. Multipart messages are always handled recursively. The handling of message/* messages are controlled by class $=s. The class $=e contains the ContentTransferEncodings that can be 8->7 bit encoded. It is predefined to contain "7bit", "8bit", and "binary". The class $=s contains the set of subtypes of message that can be treated recursively. By default it contains only "rfc822". Other "message/*" types cannot be 8->7 bit encoded. If a message containing eight-bit data is sent to a seven-bit host, and that message cannot be encoded into seven bits, it will be stripped to 7 bits.

The three classes $=U, $=Y, and $=Z are defined to describe the hosts requiring the use of a uucp mailer. Specifically, $=U should contain all hosts requiring the uucp-old mailer. $=Y should contain all hosts requiring the uucp-new mailer. Finally, $=Z should contain all hosts requiring the uucp-uudom mailer. Each uucp host should belong to one of these classes.

Sendmail can be compiled to allow a scanf(3) string on the F line. This lets you do simplistic parsing of text files. For example, to read all the user names in your system /etc/passwd file into a class, use:

FL/etc/passwd %[^:]

which reads every line up to the first colon.

Changing the Host Name:  $w contains all the possible names for the local host. It defines aliases. $w specifies the name and all aliases for your host system. If your system uses different names for two different network connections, enter both names as part of the host name class. If you do not define both names, mail sent to the undefined name is returned to the sender.

$w alias aliasn...

By default, the sendmail command reads what has been set with the hostname command and uses it to initialize the host and domain name macros and classes. Change the configuration file macros only if you want the sendmail host and domain names to be different from those set by the hostname command.

To change the host name:
1. Enter the command:
   vi /etc/mail/sendmail.cf
2. Find the lines beginning with Dj and Dw. Dj and Dw override the host and domain names set with "hostname".
3. Replace Dj and Dw with the new hostname information. For example, if your hostname is brown.newyork.abc.com, and you have one alias, brown2, enter:
4. Save the file and exit the editor.

Creating a Class Using a File:  To define a class whose members are listed in an external file (one member per line), use a control line that begins with the letter F. The syntax for the F class definition is:

FClass FileName [Format]

Class is the name of the class that matches any of the words listed in FileName. Filename is the full path name of file (for convenience, you may wish to put the file in the /etc/mail directory). Format is an optional scanf subroutine format specifier that indicates the format of the elements of the class in FileName. The Format specifier can contain only one conversion specification.

M — Define Mailer:  Programs and interfaces to mailers are defined in this line. The format is:

Mname, {field=value}*

where name is the name of the mailer (used internally only) and the "field=arguments" pairs define attributes of the mailer. Fields are:
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path</td>
<td>The path name of the mailer</td>
</tr>
<tr>
<td>Flags</td>
<td>Special flags for this mailer</td>
</tr>
<tr>
<td>Sender</td>
<td>Rewrite set(s) for sender addresses</td>
</tr>
<tr>
<td>Recipient</td>
<td>Rewrite set(s) for recipient addresses</td>
</tr>
<tr>
<td>Argv</td>
<td>An argument vector to pass to this mailer</td>
</tr>
<tr>
<td>Eol</td>
<td>The end-of-line string for this mail</td>
</tr>
<tr>
<td>Maxsize</td>
<td>The maximum message length to this mailer</td>
</tr>
<tr>
<td>maxmessages</td>
<td>The maximum message delivers per connection</td>
</tr>
<tr>
<td>Linelimit</td>
<td>The maximum line length in the message body</td>
</tr>
<tr>
<td>Directory</td>
<td>The working directory for the mailer</td>
</tr>
<tr>
<td>Userid</td>
<td>The default user and group ID to run</td>
</tr>
<tr>
<td>Nice</td>
<td>The nice(2) increment for the mailer</td>
</tr>
<tr>
<td>Charset</td>
<td>The default character set for 8-bit characters</td>
</tr>
<tr>
<td>Type</td>
<td>The MTS type information (used for error messages)</td>
</tr>
<tr>
<td>Wait</td>
<td>The maximum time to wait for the mailer</td>
</tr>
<tr>
<td>/</td>
<td>The root directory for the mailer</td>
</tr>
</tbody>
</table>

Only the first character of the field name is checked.

The flags in the following list may be set in the mailer description. Any other flags may be used freely to conditionally assign headers to messages destined for particular mailers. Flags marked with `-` are not interpreted by the `sendmail` binary; these are conventionally used to correlate to the flags portion of the H line. Flags marked with `=` apply to the mailers for the sender address rather than the usual recipient mailers.

<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>Run Extended SMTP (ESMTP) protocol (defined in RFCs 1651, 1652, and 1653). This flag defaults on if the SMTP greeting message includes the word “ESMTP”.</td>
</tr>
<tr>
<td>A</td>
<td>Look up the user part of the address in the alias database. Normally this is only set for local mailers.</td>
</tr>
<tr>
<td>b</td>
<td>Force a blank line on the end of a message. This is intended to work around some versions of <code>/bin/mail</code> that require a blank line, but do not provide it themselves. It would not normally be used on network mail.</td>
</tr>
<tr>
<td>c</td>
<td>Do not include comments in addresses. This should only be used if you have to work around a remote mailer that gets confused by comments. This strips addresses of the form &quot;Phrase &lt;address&gt;&quot; or &quot;address (Comment)&quot; down to just &quot;address&quot;.</td>
</tr>
</tbody>
</table>
| C=   | If mail is received from a mailer with this flag set, any addresses in the header that do not have an at sign (@) after being rewritten by ruleset three will have the " @domain" clause from the sender envelope address tacked on. This allows mail with headers of the form:

```
From: usera@hosta
To: userb@hostb, userc
```

...to be rewritten automatically (although not reliably) as:

```
From: usera@hosta
To: userb@hostb, userc@hosta
```

d | Do not include angle brackets around route-address syntax addresses. This is useful on mailers that are going to pass addresses to a shell that might interpret angle brackets as I/O redirection. |
| D-   | This mailer wants a "Date:" header line. |
| e    | This mailer is expensive to connect to, so try to avoid connecting normally. Any necessary connection will occur during a queue run. |
| E    | Escape lines beginning with "From" in the message with a `>` sign. |
| f    | The mailer wants a `-f` from flag, but only if this is a network forward operation (that is, the mailer will give an error if the executing user does not have special permissions). |
| F-   | This mailer wants a "From:" header line. |
Flag | Description
---|---
g | Normally, sendmail sends internally generated error messages using the null return address as required by RFC 1123. However, some mailers do not accept a null return address. If necessary, you can set the g flag to prevent sendmail from obeying the standards; error messages will be sent as from the MAILER-DAEMON (actually, the value of the $n macro).
h | Uppercase should be preserved in host names for this mailer.
i | Do User Database rewriting on envelope sender address.
l | This mailer will be speaking SMTP to another sendmail, as such it can use special protocol features. This option is not required (that is, if this option is omitted the transmission will still operate successfully, although perhaps not as efficiently as possible).
j | Do User Database rewriting on recipients as well as senders.
k | Normally when sendmail connects to a host via SMTP, it checks to make sure that this is not accidentally the same host name as might happen if sendmail is misconfigured or if a long-haul network interface is set in loopback mode. This flag disables the loopback check. It should only be used under very unusual circumstances.
K | Currently unimplemented. Reserved for chunking.
L | This mailer is local (that is, final delivery will be performed).
M- | This mailer wants a “Message-id:” header line.
n | Do not insert a UNIX-style “From” line on the front of the message.
o | Always run as the owner of the recipient mailbox. Normally sendmail runs as the sender for locally generated mail or as “daemon” (actually, the user specified in the u option) when delivering network mail. The normal behavior is required by most locally mailers, which will not allow the envelope sender address to be set unless the mailer is running as daemon. This flag is ignored if the S flag is set.
p | Use the route-addr style reverse-path in the SMTP “MAIL FROM:” command rather than just the return address; although this is required in RFC821 section 3.1, many hosts do not process reverse-paths properly. Reverse-paths are officially discouraged by RFC 1123.
q | When an address that resolves to this mailer is verified (SMTP VRFY command), generate 250 responses instead of 252 responses. This will imply that the address is local.
r | Same as f, but sends an -r flag.
R | Open SMTP connections from a “secure” port. Secure ports are not secure except on UNIX machines, so it is unclear that this adds anything.
s | Strip quote characters (‘ and \) off the address before calling the mailer.
S | Do not reset the userid before calling the mailer. This would be used in a secure environment where sendmail ran as root. This could be used to avoid forged addresses. If the U= field is also specified, this flag causes the userid to always be set to that user and group (instead of leaving it as root).
u | Uppercase should be preserved in host names for this mailer.
U | This mailer wants UUCP-style “From” lines with the “remote from <host>” on the end.
w | The user must have a valid account on this machine (getpwnam must succeed). If not, the mail is bounced. This is required to get “.forward” capability.
x- | This mailer wants a “Full-Name:” header line.
X | This mailer wants to use the hidden dot algorithm as specified in RFC821; basically, any line beginning with a dot will have an extra dot prepended (to be stripped at the other end). This ensures that lines in the message containing a dot will not terminate the message prematurely.
z | Run Local Mail Transfer Protocol (LMTP) between sendmail and the local mailer. This is a variant on SMTP defined in RFC 2033 that is specially designed for delivery to a local mailbox.
0 | Do not look up Mx records for hosts via SMTP.
3 | Extend the list of characters converted to =XX notation when converting to Quoted-Printable to include those that do not map cleanly between ASCII and EBCDIC. Useful if you have IBM mainframes on site.
5 | If no aliases are found for this address, pass the address through ruleset 5 for possible alternate resolution. This is intended to forward the mail to an alternate delivery spot.
6 | Strip headers to seven bits.
Strip all output to seven bits. This is the default if the L flag is set. Note that clearing this option is not sufficient to get full eight-bit data passed through sendmail. If the 7 option is set, this is essentially always set, because the eighth bit was stripped on input. Note that this option will only impact messages that did not have 8->7 bit MIME conversions performed.

If set, it is acceptable to send eight bit data to this mailer; the usual attempt to do 8->7 bit MIME conversions will be bypassed.

If set, do limited 7->8 bit MIME conversions. These conversions are limited to text/plain data.

Check addresses to see if they begin with a `|'. If they do, convert them to the “prog” mailer.

Check addresses to see if they begin with a `/'. If they do, convert them to the “*file*” mailer.

Look up addresses in the user database.

Do not attempt delivery on initial recipient of a message or on queue runs unless the queued message is selected using one of the -ql/-qR/-qS queue run modifiers or an ETRN request.

Note: Configuration files prior to level 6 assume the `A', `w', `5', `:', `|', `/', and `@' options on the mailer named “local”.

The mailer with the special name “error” can be used to generate a user error. The (optional) host field is an exit status to be returned, and the user field is a message to be printed. The exit status may be numeric or one of the values USAGE, NOUSER, NOHOST, UNAVAILABLE, SOFTWARE, TEMPFAIL, PROTOCOL, or CONFIG to return the corresponding EX_ exit code. For example, the entry:

```
$#error $@ NOHOST $: Host unknown in this domain
```

on the RHS of a rule will cause the specified error to be generated and the “Host unknown” exit status to be returned if the LHS matches. It is always available for use in O, S, and check_ ... rulesets and it cannot be defined with M commands.

The mailer named “local” must be defined in every configuration file. This is used to deliver local mail, and is treated specially in several ways. Additionally, three other mailers named “prog”, “*file*”, and “*include*” may be defined to tune the delivery of messages to programs, files, and :include: lists respectively. They default to:

```
Mprog, P=/bin/sh, F=lsodq9, T=DNS/RFC822/X-Unix, A=sh -c $u
M*file*, P=[[FILE]], F=lsodpq9, T=DNS/RFC822/X-Unix, A=FILE $u
M*include*, P=/dev/null, F=su, A=INCLUDE $u
```

The Sender and Recipient rewrite sets may either be a simple ruleset ID or may be two IDs separated by a slash. If so, the first rewrite set is applied to envelope addresses, and the second is applied to headers. Setting any value to zero disables the corresponding mailer-specific rewriting.

The Directory field is a path of directories to try. For example, the definition D=$z:/$ tries to execute the recipient’s home directory, but if that is not available, it tries to execute in the root of the filesystem. Use this on the prog mailer only, because some shells (e.g., csh) do not execute if they cannot read the home directory. Because the queue directory usually cannot be read by unauthorized users, csh scripts can fail if they are used as recipients.

The Userid field specifies the default user and group ID to run. It overrides the DefaultUser option q.v. If the S mailer flag is also specified, the user and group ID will run in all circumstances. Use the form user:group to set both the user and group ID. Either of these variables may be an integer or a symbolic name that is looked up in the passwd and group files respectively.

The Charset field is used when converting a message to MIME. It is the character set used in the Content-Type: header. If it is not set, the DefaultCharset option is used. If the DefaultCharset is not set, the value unknown-8bit is used. The Charset field applies to the sender’s mailer; not the recipient’s mailer.
For example: if the envelope sender address is on the local network and the recipient is on an external network, the character set is set from the Charset= field for the local network mailer, not the external network mailer.

The Type field sets the type of information used in MIME error messages (as defined by RFC 1984). It contains three values that are separated by slashes: the MTA type (a description of how hosts are named), address type (a description of e-mail addresses), and diagnostic type (a description of error diagnostic codes). Each must be a registered value or begin with X-. The default is dns/rfc822/smtp.

**Mailer Specifications Examples:**
1. To specify a local delivery mailer enter:
   
   `Mlocal, P=/usr/bin/bellmail, F=lsDFMmn, S=10, R=20, A@mail $u`

   The mailer is called `local`. Its path name is `/usr/bin/bellmail`. The mailer uses the following flags:

   - l: Specifies local delivery.
   - s: Strips quotation marks from addresses.
   - DFM: Requires Date:, From:, and Message-ID: fields.
   - m: Delivers to multiple users.
   - n: Does not need an operating system From line at the start of the message.

   Rule set 10 should be applied to sender addresses in the message. Rule set 20 should be applied to recipient addresses. Additional information sent to the mailer in the A field is the word `mail` and words containing the recipient's name.

   **H — Define Header:** The format of the header lines that `sendmail` inserts into the message are defined by the H line. The syntax of this line is one of the following:

   - `H hname:htemplate`
   - `H?[mflags?]hname:htemplate`
   - `H?[${macro}]hname:htemplate`

   Continuation lines in this spec are reflected directly into the outgoing message. The htemplate is macro expanded before insertion into the message. If the mflags (surrounded by question marks) are specified, at least one of the specified flags must be stated in the mailer definition for this header to be automatically output. If one of these headers is in the input, it is reflected to the output regardless of these flags.

   Some headers have special semantics that will be described later.

   A secondary syntax allows validation of headers as they being read. To enable validation, use:

   - `HHeader: $>Ruleset`
   - `HHeader: $>+Ruleset`

   The indicated `Ruleset` is called for the specified `Header`. Like other `check_*` rulesets, it can return `$#error` to reject the message or `$#discard` to discard the message. The header is treated as a structured field, so comments (in parentheses) are deleted before processing, unless the second form `$>+` is used.

   For example, the following configuration lines:

   - `HMessage-Id: $>CheckMessageId`
   - `SCheckMessageId`
   - `R<=$+G$+ $@OK`
   - `R$* $#error $: Illegal Message-Id header`

   would refuse any message header that had a Message-Id: header of any of the following forms:
Message-Id: <>
Message-Id: some text
Message-Id: <legal test@domain> extra text

**Message Headings in the sendmail.cf File:** Lines in the configuration file that begin with a capital letter \textit{H}, define the format of the headers used in messages. The format of the H command is:

Lines in the configuration file that begin with a capital letter \textit{H}, define the format of the headers used in messages. The format of the H control line is:

\texttt{H[?MailerFlags?]FieldName: Content}

The variable parameters are defined as:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>MailerFlags</td>
<td>Determines whether the H line is used. This parameter is optional. If you supply this parameter, surround it with ? (question marks). If the mailer requires the field defined by this control line (as indicated in the mailer definition’s flags field), then the H control line is included when formatting the heading. Otherwise, the H control line is ignored.</td>
</tr>
<tr>
<td>FieldName</td>
<td>Contains the text displayed as the name of the field in the heading information. Typical field names include \texttt{From:}, \texttt{To:}, and \texttt{Subject:}.</td>
</tr>
<tr>
<td>Content</td>
<td>Defines the information that is displayed following the field name. Usually macros specify this information.</td>
</tr>
</tbody>
</table>

These example lines are from a typical /etc/mail/sendmail.cf file:

**Example**

**Meaning**

\texttt{H??Return-Path: <$g>}

Defines a field called Return-Path that displays the content of the \texttt{$g} macro (sender address relative to the recipient). The ???? portion indicates this line is only used if the mailer uses the \texttt{P} flag (the mailer requires a Return-Path line). The header is generated only if the mailer has the indicated flag. If the header appears in the input message, it is passed through unchanged.

\texttt{HReceived: $?sfrom $s $by $j ($v/$Z) id $i; $b}

Defines a field called Received. This field includes:

- \texttt{$?sfrom $s $}
- \texttt{$by $j}
- \texttt{($v/$Z) id $i; $b}

- Displays the text \texttt{from} followed by the content of the \texttt{$s} macro if an \texttt{s} macro is defined (sender’s host name).
- Displays the text \texttt{by} followed by the content of the \texttt{$j} macro (official name for a specific location).
- Displays the version of the \texttt{sendmail} command (\texttt{$v}) and the version of the /etc/mail/sendmail.cf file (\texttt{$Z}), set off by parentheses and separated by a slash.
- Displays the text \texttt{id} followed by the content of the \texttt{$i} macro (mail-queue ID of the message) and a ; (semicolon).
- Displays the current date.

**O — Set Option:** There are several global options that can be set from a configuration file. The syntax of this line is:

\texttt{O option=value}

This sets option \texttt{equal} to \texttt{value}. The options supported are listed in the following table.
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
</table>
| **AliasFile**={spec, spec, ...} | Specify possible alias file(s). Each spec should be in the format class:: file where class:: is optional and defaults to **implicit** if not included. Depending on how **sendmail** is compiled, valid classes are:  
  implicit  
    search through a compiled-in list of alias file types, for back compatibility  
  hash  
    if NEWDB is specified  
  dbm  
    if NDBM is specified  
  stab  
    Internal symbol table. Not normally used unless there is no other database lookup  
  nis  
    if NIS is specified  

If a list of specs are provided, **sendmail** searches them in order. |
| **AliasWait=time-out** | Waits up to time-out (units default to minutes) for an @:@ entry to exist in the alias database before starting up. If it does not appear in the time-out interval and the **AutoRebuildAliases** option is also set, rebuild the database. Otherwise, issue a warning. |
| **AllowBogusHELO** | Allows HELO SMTP commands that do not include a host name. Setting this violates RFC 1123 section 5.2.5, but is necessary to interoperate with several SMTP clients. If there is a value, it is still checked for legitimacy. |
| **BlankSub=c** | Sets the blank substitution character to c. Unquoted spaces in addresses are replaced by this character. If not defined, it defaults to a space and no replacement is made. |
| **CACERTPath** | Path to directory with certificates of CAs. |
| **CACERTFile** | File containing one CA certificate. |
| **CheckAliases** | Validate the RHS of aliases when rebuilding the alias database. |
| **CheckpointInterval=N** | Defines the queue checkpoint interval to every N addresses sent. If not specified, the default is 10. If your system crashes during delivery to a large list, this prevents retransmission to any but the last recipients. |
| **ClassFactor=fact** | The indicated factor is multiplied by the message class and subtracted from the priority. The message class is determined by the Precedence: field in the user header and the P lines in the configuration file. Messages with a higher Priority: will be favored. If not specified, the defaults is 1800. |
| **ClientCertFile** | The file containing the certificate of the client. This certificate is used when **sendmail** acts as client. |
Option

**ClientPortOptions**=options

Sets client SMTP options. The options are *key=value* pairs separated by commas. Known keys are:

- **Port**  Name/number of source port for connection. Default is any free port.
- **Addr**  Address mask. Default is \texttt{INADDR\_ANY}. Can be a numeric address in dot notation or a network name.
- **Family**  Address family. Default is \texttt{INET}.
- **SndBufSize**  Size of TCP send buffer.
- **RcvBufSize**  Size of TCP receive buffer.

**Modifier**

Flags for the daemon. Can be the following character:

- **h**  Use name of interface for \texttt{HELO} command

If \texttt{h} is set, the name corresponding to the outgoing interface address (whether chosen via the \texttt{Connection} parameter or the default) is used for the \texttt{HELO/EHLO} command.

**ClientKeyFile**

The file containing the private key belonging to the client certificate.

**ColonOkInAddr**

If set, colons are acceptable in e-mail addresses, for example:

```
host:user
```

If not set, colons indicate the beginning of a RFC 822 group construct, illustrated below:

```
groupname: member1, member2, ... memberN;
```

Doubled colons are always acceptable, such as in

```
nodename::user
```

and proper routeaddr nesting is understood, for example:

```
<@relay:user@host>
```

This option defaults to \texttt{on} if the configuration version level is less than 6, for backward compatibility. However, it must be set to \texttt{off} for full compatibility with RFC 822.

**ConnectionCacheSize**=\texttt{N}

\(N\) is the maximum number of open connections that will be cached at a time. If not specified, the default is 1. This delays closing the current connection until either this invocation of \texttt{sendmail} connects to another host or it terminates. Setting it to 0 causes connections to closed immediately. Because this consumes file descriptors, the connection cache should be kept small: 4 is a practical maximum.

**ConnectionCacheTimeout**=\texttt{time-out}

\textit{Timeout} is the maximum amount of time a cached connection will be permitted to be idle. If this time is exceeded, the connection is immediately closed. This value should be small: 10 minutes is a practical maximum; the default is 5 minutes. Before \texttt{sendmail} uses a cached connection, it always sends a \texttt{RSET} command to check the connection. If this fails, it reopens the connection. This keeps your end from failing if the other end times out.

**ConnectOnlyTo**=\texttt{address}

Can be used to override the connection address for testing purposes.

**ConnectionRateThrottle**=\texttt{N}

If set, allows no more than \(N\) incoming daemon connections in a one second period. This is intended to flatten peaks and allow the load-average checking to cut in. If not specified, the default is 0 (no limits).
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ControlSocketName=name</strong></td>
<td>Defines the name of the control socket for daemon management. A running <code>sendmail</code> daemon can be controlled through this named socket. Available commands are: <strong>help</strong>, <strong>restart</strong>, <strong>shutdown</strong>, and <strong>status</strong>. The <strong>status</strong> command returns the current number of daemon children, the maximum number of daemon children, free disk space blocks of the queue directory, and the load average of the machine expressed as an integer. If not set, no control socket will be available.</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>DaemonPortOptions=options</code></td>
<td>Set server SMTP options. Each instance of <code>DaemonPortOptions</code> leads to an additional incoming socket. The options are key=value pairs. Known keys are:</td>
</tr>
<tr>
<td>Name</td>
<td>User-definable name for the daemon. Default is <code>Daemon</code>. Is used for error messages and logging.</td>
</tr>
<tr>
<td></td>
<td>To be able to send mail but not receive mail on a system, edit <code>/etc/mail/sendmail.cf</code>:</td>
</tr>
<tr>
<td></td>
<td># 0 DaemonPortOptions=Name=MTA</td>
</tr>
<tr>
<td></td>
<td>0 DaemonPortOptions=NAME=NoMTA4, Family=inet, Addr=127.0.0.1</td>
</tr>
<tr>
<td>Port</td>
<td>Name/number of listening port. Default is smtp.</td>
</tr>
<tr>
<td>Addr</td>
<td>Address mask. Default is INADDR_ANY. This may be a numeric address in dot notation or a network name.</td>
</tr>
<tr>
<td>Family</td>
<td>Address family. Default is INET (IPv4). IPv6 systems that need to accept IPv6 connections, should add additional Family=inet6 DaemonPort Options lines.</td>
</tr>
<tr>
<td>Listen</td>
<td>Size of listen queue. Default is 10.</td>
</tr>
<tr>
<td>Modifier</td>
<td>Flags for the daemon. Can be a sequence, without delimiters, of the following characters:</td>
</tr>
<tr>
<td>a</td>
<td>Always require authentication.</td>
</tr>
<tr>
<td>b</td>
<td>Bind to interface through which mail has been received for the outgoing connection.</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong>: Use the b flag only if outgoing mail can be routed through the incoming connection's interface to its destination. No attempt is made to catch problems that result from incorrectly configuring this parameter. It should only be used for virtual hosting where each virtual interface can connect to every possible location. The b flag can override the settings through ClientPort Options. In addition, sendmail will listen on a new socket for each occurrence of the <code>DaemonPortOptions</code> subcommand in a configuration file.</td>
</tr>
<tr>
<td>c</td>
<td>Perform hostname canonicalization (.cf). Can change the default for hostname canonicalization in the <code>sendmail.cf</code> file. See the documentation for <code>FEATURE(nocanonify)</code> in the <code>/user/samples/tcpip/sendmail/README</code> file.</td>
</tr>
<tr>
<td>f</td>
<td>Require fully qualified hostname (.cf). Cannot use addresses in the form <code>user@host</code> unless they are directly submitted.</td>
</tr>
<tr>
<td>u</td>
<td>Allow unqualified addresses (.cf) (including sender addresses).</td>
</tr>
<tr>
<td>C</td>
<td>Do not perform hostname canonicalization. Can change the default for hostname canonicalization in the <code>sendmail.cf</code> file. See the documentation for <code>FEATURE(nocanonify)</code> in the <code>/user/samples/tcpip/sendmail/README</code> file.</td>
</tr>
<tr>
<td>E</td>
<td>Do not allow ETRN (see RFC 2476).</td>
</tr>
</tbody>
</table>

One way to specify a message submission agent (MSA) that always require authentication is:

0 DaemonPortOptions=Name=MSA,Port=587,M=Ea

Modifiers marked with (.cf) are effective only when used in the standard configuration file (available through `$(${daemon_flags})`) and cannot be used from the command line.
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DaemonPortOptions=</td>
<td>Options cont.</td>
</tr>
<tr>
<td>SndBufSize</td>
<td>Size of TCP send buffer.</td>
</tr>
<tr>
<td>RcvBufSize</td>
<td>Size of TCP receive buffer</td>
</tr>
<tr>
<td>DefaultAuthInfo</td>
<td>Filename that contains default authentication information for outgoing connections. This file must contain the user ID, authorization ID, password (plain text), and the realm to use on separate lines and must be readable only by root (or the trusted user). If no realm is specified, $j$ is used.</td>
</tr>
<tr>
<td>DefaultCharSet=charset</td>
<td>When a message that has 8-bit characters, but is not in MIME format, is converted to MIME (see the EightBitMode option) a character set must be included in the Content-Type: header. This character set is normally set from the CharSet= field of the mailer descriptor. If that is not set, the value of this option is used. If this option is not set, the value unknown-8bit is used.</td>
</tr>
<tr>
<td>DataFileBufferSize=</td>
<td>Sets threshold in bytes before a memory-based queue data file becomes disk-based. The default is 4096 bytes.</td>
</tr>
<tr>
<td>DeadLetterDrop=file</td>
<td>Defines the location of the systemwide dead.letter file, formerly hardcoded to /usr/tmp/dead.letter. If this option is not set, sendmail will not attempt to save to a systemwide dead.letter file in the event it cannot bounce the mail to the user or postmaster. Instead, it will rename the qf file.</td>
</tr>
<tr>
<td>DefaultUser=user:group</td>
<td>Set the default user ID for mailers to user:group. If group is omitted and user is a user name (as opposed to a numeric user ID) the default group listed in the /etc/passwd file for that user is used as the default group. Both user and group may be numeric. Mailers without the S flag in the mailer definition will run as this user. When not specified, the default is 1:1. The value can also be given as a symbolic user name.</td>
</tr>
<tr>
<td>DeliveryMode=x</td>
<td>Deliver in mode x. Legal modes are:</td>
</tr>
<tr>
<td></td>
<td>i Deliver interactively (synchronously).</td>
</tr>
<tr>
<td></td>
<td>b Deliver in background (asynchronously).</td>
</tr>
<tr>
<td></td>
<td>q Just queue the message (deliver during queue run).</td>
</tr>
<tr>
<td></td>
<td>d Defer delivery and all map lookups (deliver during queue run).</td>
</tr>
<tr>
<td>DialDelay=sleeptime</td>
<td>Dial-on-demand network connections can see time-outs if a connection is opened before the call is set up. If this is set to an interval and a connection times out on the first connection being attempted, sendmail will sleep for this amount of time and try again. This should give your system time to establish the connection to your service provider. Units default to seconds, so DialDelay=5 would use a five second delay. If not specified, the default is 0 (no retry).</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DontBlameSendmail</td>
<td>In order to avoid possible cracking attempts caused by world- and group-writable files and directories, <em>sendmail</em> does paranoid checking when opening most of its support files. However, if a system must run with a group-writable /etc directory, then this checking must be turned off. Note that turning off this checking will make your system more vulnerable to attack. The arguments are individual options that turn off checking:</td>
</tr>
<tr>
<td>Safe</td>
<td>No special handling.</td>
</tr>
<tr>
<td>AssumeSafeChown</td>
<td>Assume that the chown call is restricted to root. Because some systems are set up to permit regular users to give away their files to other users on some file systems, <em>sendmail</em> often cannot assume that a given file was created by the owner, particularly when it is in a writable directory. You can set this flag if you know that file giveaway is restricted on your system.</td>
</tr>
<tr>
<td>ClassFileInUnsafeDirPath</td>
<td>When reading class files (using the F line in the configuration file), allow files that are in unsafe directories.</td>
</tr>
<tr>
<td>DontWarnForwardFileInUnsafeDirPath</td>
<td>Prevent logging of unsafe directory path warnings for nonexistent forward files.</td>
</tr>
<tr>
<td>ErrorHeaderInUnsafeDirPath</td>
<td>Allow the file named in the ErrorHeader option to be in an unsafe directory.</td>
</tr>
<tr>
<td>FileDeliveryToHardLink</td>
<td>Allow delivery to files that are hard links.</td>
</tr>
<tr>
<td>FileDeliveryToSymLink</td>
<td>Allow delivery to files that are symbolic links.</td>
</tr>
<tr>
<td>ForwardFileInUnsafeDirPath</td>
<td>Allow .forward files in unsafe directories.</td>
</tr>
<tr>
<td>ForwardFileInUnsafeDirPathSafe</td>
<td>Allow .forward files that are in an unsafe directory to include references to program and files.</td>
</tr>
<tr>
<td>ForwardFileInGroupWritableDirPath</td>
<td>Allow .forward files in group writable directories.</td>
</tr>
<tr>
<td>GroupWritableAliasFile</td>
<td>Allow group-writable alias files.</td>
</tr>
<tr>
<td>GroupWritableDirPathSafe</td>
<td>Change the definition of unsafe directory to consider group-writable directories to be safe. World-writable directories are always unsafe.</td>
</tr>
<tr>
<td>GroupWritableForwardFileSafe</td>
<td>Accept group-writable .forward files.</td>
</tr>
<tr>
<td>GroupWritableIncludeFileSafe</td>
<td>Accept group-writable :include: files.</td>
</tr>
<tr>
<td>HelpFileInUnsafeDirPath</td>
<td>Allow the file named in the HelpFile option to be in an unsafe directory.</td>
</tr>
<tr>
<td>IncludeFileInUnsafeDirPath</td>
<td>Allow :include: files in unsafe directories.</td>
</tr>
</tbody>
</table>

Chapter 1. System Files  253
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IncludeFileInUnsafeDirPathSafe</td>
<td>Allow .forward files that are in an unsafe directory to include references to program and files.</td>
</tr>
<tr>
<td>IncludeFileInGroupWritableDirPath</td>
<td>Allow :include: files in group writable directories.</td>
</tr>
<tr>
<td>InsufficientEntropy</td>
<td>Try to use STARTTLS even if the PRGN for OpenSSL is not properly seeded despite the security problems.</td>
</tr>
<tr>
<td>LinkedAliasFileInWritableDir</td>
<td>Allow alias files that are links in a writable directory.</td>
</tr>
<tr>
<td>LinkedClassFileInWritableDir</td>
<td>Allow class files that are links in writable directories.</td>
</tr>
<tr>
<td>LinkedForwardFileInWritableDir</td>
<td>Allow .forward files that are links in writable directories.</td>
</tr>
<tr>
<td>LinkedIncludeFileInWritableDir</td>
<td>Allow :include: files that are links in writable directories.</td>
</tr>
<tr>
<td>LinkedMapInWritableDir</td>
<td>Allow map files that are links in writable directories.</td>
</tr>
<tr>
<td>LinkedServiceSwitchFileInWritableDir</td>
<td>Allow the service switch file to be a link even if the directory is writable.</td>
</tr>
<tr>
<td>MapInUnsafeDirPath</td>
<td>Allow maps (such as hash, btree, and dbm files) in unsafe directories.</td>
</tr>
<tr>
<td>NonRootSafeAddr</td>
<td>Do not mark file and program deliveries as unsafe if sendmail is not running.</td>
</tr>
<tr>
<td>RunProgramInUnsafeDirPath</td>
<td>Run programs that are in writable directories.</td>
</tr>
<tr>
<td>RunWritableProgram</td>
<td>Run programs that are group- or world-writable.</td>
</tr>
<tr>
<td>TrustStickyBit</td>
<td>Allow group- or world-writable directories if the sticky bit is set on the directory. Do not set this on systems which do not honor the sticky bit on directories.</td>
</tr>
<tr>
<td>WorldWritableAliasFile</td>
<td>Accept world-writable alias files.</td>
</tr>
<tr>
<td>WriteMapToHardLink</td>
<td>Allow writes to maps that are hard links.</td>
</tr>
<tr>
<td>WriteMapToSymLink</td>
<td>Allow writes to maps that are symbolic links.</td>
</tr>
<tr>
<td>WriteStatsToHardLink</td>
<td>Allow the status file to be a hard link.</td>
</tr>
<tr>
<td>WriteStatsToSymLink</td>
<td>Allow the status file to be a symbolic link.</td>
</tr>
</tbody>
</table>

**Safe** is the default. The details of these flags are described above. Use of this option is not recommended.
Option Description

DontExpandCnames  The standards say that all host addresses used in a mail message must be fully canonical. For example, if your host is named `Cruft.Foo.ORG` and also has an alias of `FTP.Foo.ORG`, the name `Cruft.Foo.ORG` must be used at all times. This is enforced during host name canonicalization (`[$ ... $]` lookups). If this option is set, the protocols will be ignored and the wrong name will be used. However, the IETF is moving toward changing this standard, so the behavior may become acceptable. Please note that hosts downstream may still rewrite the address to be the true canonical name.

DontInitGroups  If set, `sendmail` will avoid using the `initgroups(3)` call. If you are running NIS, this causes a sequential scan of the `groups.byname` map, which can cause your NIS server to be badly overloaded in a large domain. The cost of this is that the only group found for users will be their primary group (the one in the password file), which will make file access permissions somewhat more restrictive. Has no effect on systems that do not have group lists.

DontProbeInterfaces  `Sendmail` normally finds the names of all interfaces active on your machine when it starts up and adds their name to the `$=w` class of known host aliases. If you have a large number of virtual interfaces or if your DNS inverse lookups are slow this can be time consuming. This option turns off that probing. However, you will need to be certain to include all variant names in the `$=w` class by some other mechanism.

DontPruneRoutes  `Sendmail` tries to eliminate any unnecessary explicit routes when sending an error message (as discussed in RFC 1123 S 5.2.6). For example, when sending an error message to `<@known1,@known2,@known3:user@unknown>`, `sendmail` will strip off the @known1,@known2 in order to make the route as direct as possible. However, if the RR option is set, this will be disabled, and the mail will be sent to the first address in the route, even if later addresses are known. This may be useful if you are caught behind a firewall.

DoubleBounceAddress  If an error occurs when sending an error message, send the error report to the indicated address. This is termed a double bounce because it is an error bounce that occurs when trying to send another error bounce. The address is macro expanded at the time of delivery. If not set, it defaults to `postmaster`.

EightBitMode  Set handling of eight-bit data. There are two kinds of eight-bit data:

- Data declared as eight-bit using the `BODY=8BITMIME` ESMTP declaration or the `-B8BITMIME` command line flag
- Undeclared 8-bit data, which is input that just happens to be eight bits.

There are three basic operations that can happen:

- Undeclared 8-bit data can be automatically converted to 8BITMIME.
- Undeclared 8-bit data can be passed as-is, without conversion to MIME.
- Declared 8-bit data can be converted to 7-bits for transmission to a non-8BITMIME mailer.

Possible actions are:

- `s`  Reject undeclared 8-bit data (strict).
- `m`  Convert undeclared 8-bit data to MIME (mime).
- `p`  Pass undeclared 8-bit data (pass).

In all cases properly declared 8BITMIME data will be converted to 7BIT as needed.
**ErrorHeader=file-or-message**

Prepend error messages with the indicated message. If it begins with a slash (/), it is assumed to be the pathname of a file containing a message, which is the recommended setting. Otherwise, it is a literal message. The error file might contain the name, e-mail address, and/or phone number of a local postmaster who could provide assistance to end users. If the option is missing or null, or if it names a file which does not exist or are not readable, no message is printed.

**ErrorMode=x**

Dispose of errors using mode x. The values for x are:

- **p** Print error messages (default).
- **q** No messages, just give exit status.
- **m** Mail back errors.
- **w** Write back errors (mail if user not logged in).
- **e** Mail back errors and give zero exit status always.

**.IlbackMXhost=fallbackhost**

If specified, the fallbackhost acts like a very low priority MX on every host. This is intended to be used by sites with poor network connectivity. Messages which are undeliverable due to temporary address failures, such as in a DNS failure, also go to the FallBackMX host.

**ForkEachJob**

If set, deliver each job that is run from the queue in a separate process. Use this option if you are short of memory, because the default tends to consume considerable amounts of memory while the queue is being processed.

**ForwardPath=path**

Sets the path for searching for users’ .forward files. The default is $z/.forward. Some sites that use the automounter may prefer to change this to /var/forward/$u to search a file with the same name as the user in a system directory. It can also be set to a sequence of paths separated by colons. Sendmail stops at the first file it can successfully and safely open. For example, /var/forward/$u:$z/.forward will search first in /var/forward/username and then in ~username/.forward, but only if the first file does not exist.

**HelpFile=file**

Specifies the help file for SMTP. If no file name is specified, helpfile is used.

**HoldExpensive**

If an outgoing mailer is marked as being expensive, do not connect immediately. This requires that queueing be compiled in, because it will depend on a queue run process to actually send the mail.

**HostsFile=path**

Specifies the path to the hosts database, normally /etc/hosts. This option is only consulted when sendmail is canonicalizing addresses, and then only when files is in the hosts service switch entry. In particular, this file is never used when looking up host addresses; that is under the control of the system gethostbyname(3) routine.

**HostStatusDirectory=path**

Sets the location of the long term host status information. When set, information about the status of hosts (such as if the host down or not accepting connections) will be shared between all sendmail processes. Normally, this information is only held within a single queue run. This option requires a connection cache of at least 1 to function. If the option begins with a leading /, it is an absolute pathname; otherwise, it is relative to the mail queue directory. A suggested value for sites desiring persistent host status is .hoststat, which is a subdirectory of the queue directory.

**IgnoreDots**

Ignore dots in incoming messages. This is always disabled when reading SMTP mail, and as a result, dots are always accepted.

**LDAPDefaultSpec=spec**

Sets a default map specification for LDAP maps. The value should only contain LDAP specific settings such as -h host -p port -d bindDN. The settings will be used for all LDAP maps unless the individual map specification overrides a setting. This option should be set before any LDAP maps are defined.

**LogLevel=n**

Set the log level to n. Defaults to 9.

**Mxvalue**

Set the macro x to value. This is intended only for use from the command line. The -M flag is preferred.
MatchGECOS

Allow fuzzy matching on the GECOS field. If this flag is set, and the usual user name lookups fail (that is, there is no alias with this name and a `getpwnam` fails), sequentially search the password file for a matching entry in the GECOS field. This also requires that MATCHGECOS be turned on during compilation. This option is not recommended.

MaxAliasRecursion=N

N is the maximum depth of alias recursion. Default is 10.

MaxDaemonChildren=N

If set, `sendmail` will refuse connections when it has more than N children processing incoming mail or automatic queue runs. This does not limit the number of outgoing connections. If not set, there is no limit to the number of children; the system load averaging will control this.

MaxHeadersLength=N

N is the maximum length of the sum of all headers. This can be used to prevent a denial of service attack. The default is no limit.

MaxHopCount=N

The maximum hop count. Messages that have been processed more than N times are assumed to be in a loop and are rejected. Default is 25.

MaxMessageSize=N

Specify the maximum message size to be advertised in the ESMTP EHLO response. Messages larger than N will be rejected.

MaxMimeHeaderLength

Sets the maximum length of certain MIME header field values to N characters. If M is specified, certain headers that take parameters will use M instead of N. If M is not specified, these headers will use one half of N. By default, these values are 0, which indicates no checks are done.

MaxQueueRunSize=N

N is the maximum number of jobs that will be processed in a single queue run. If not set, there is no limit on the size. If you have very large queues or a very short queue run interval this could be unstable. However, because the first N jobs in queue directory order are run (rather than the N highest priority jobs) this should be set as high as possible to avoid losing jobs that happen to fall late in the queue directory.

MaxRecipientsPerMessage=N

The maximum number of recipients that will be accepted per message in an SMTP transaction. If not set, there is no limit on the number of recipients per envelope.

Note: Setting this too low can interfere with sending mail from MUAs that use SMTP for initial submission.

MeToo

Send to me too, even if I am in an alias expansion. This option is deprecated and will be removed from a future version.

MinFreeBlocks=N

Sets at least N blocks free on the file system that holds the queue files before accepting e-mail via SMTP. If there is insufficient space, `sendmail` gives a 452 response to the MAIL command and invites the sender to try again later.

MinQueueAge=age

Do not process any queued jobs that have been in the queue less than the indicated time interval. This promotes system responsiveness by processing the queue frequently without taxing the system by trying jobs too often. The default units are minutes.

MustQuoteChars=s

Sets the list of characters that must be quoted if used in a full name that is in the phrase part of a phrase `<address>` syntax. The default is `. `. The characters `@`,`;`,`:` , `(` , `)` , `[]` are always added to this list.

NoRecipientAction

The action to take when you receive a message that has no valid recipient headers, such as To:, Cc:, or Bcc:. It can be:

None Passes the message on unmodified, which violates the protocol.

Add-To Adds a To: header with any recipients it can find in the envelope (which might expose Bcc: recipients).

Add-To-Undisclosed Adds a header To: undisclosed-recipients;; to make the header legal without disclosing anything.

Add-Bcc Adds an empty Bcc: header.
OldStyleHeaders

Assume that the headers may be in old format with spaces delimit names. This actually turns on an adaptive algorithm: if any recipient address contains a comma, parenthesis, or angle bracket, it will be assumed that commas already exist. If this flag is not on, only commas delimit names. Headers are always output with commas between the names. Defaults to off.

OperatorChars=charlist

The list of characters that are considered to be operators, that is, characters that delimit tokens. All operator characters are tokens by themselves; sequences of non-operator characters are also tokens. White space characters separate tokens but are not tokens themselves. For example, AAA.BBB has three tokens, but AAA BBB has two. If not set, OperatorChars defaults to .:@[]". In addition, the characters "()<>;" are always operators. Note that OperatorChars must be set in the configuration file before any rulesets.

PidFile=filename

Sets the filename of the pid file. Default is PATHSENDMAILPID. The filename is macro-expanded before it is opened.

PostmasterCopy=postmaster

If set, copies of error messages will be sent to the named postmaster. Only the header of the failed message is sent. Errors resulting from messages with a negative precedence will not be sent. Because most errors are user problems, this is not a good idea on large sites, and may contain privacy violations. The address is macro expanded at the time of delivery. Defaults to no postmaster copies.

PrivacyOptions=opt,opt,...

Sets privacy options. These are a way of insisting on stricter adherence to the SMTP protocol. The options can be one of the following:

- public  Allow open access.
- needmailhelo  Insist on HELO or EHLO command before MAIL.
- needexpnhelo  Insist on HELO or EHLO command before EXPN.
- noexpn  Do not allow EXPN, implies noverb.
- needvrfyhelo  Insist on HELO or EHLO command before VRFY.
- novrfy  Do not allow VRFY.
- noetn  Do not allow ETRN.
- noverb  Do not allow VERB.
- restrictmailq  Restrict mailq command. If mailq is restricted, only people in the same group as the queue directory can print the queue.
- restrictqrun  Restrict -q command line flag. If queue runs are restricted, only root and the owner of the queue directory can run the queue.
- noreceipts  Do not return success DSNs.
- nobodyreturn  Do not return the body of a message with DSNs.
- goaway  Do not allow SMTP status queries. Sets all flags except noreceipts, restrictmailq, restrictqrun, noetn, and nobodyreturn.
- authwarnings  Put X-Authentication-Warning: headers in messages. Authentication Warnings add warnings about various conditions that may indicate attempts to spoof the mail system, such as using an nonstandard queue directory.
Prefix the process title shown on `ps` listings with `string`. The string will be macro processed.

Use the named `dir` as the queue directory. To use multiple queues, supply a value ending with an asterisk. For example, entering `/var/spool/mqueue/q*` will use all of the directories or symbolic links to directories beginning with `q` in `/var/spool/mqueue` as queue directories. Do not change the queue directory structure while `sendmail` is running.

Use `factor` as the multiplier in the map function to decide when to just queue up jobs rather than run them. This value is divided by the difference between the current load average and the load average limit (`QueueLA` option) to determine the maximum message priority that will be sent. Default is 600000.

When the system load average exceeds `LA`, just queue messages, do not try to send them. Defaults to 8 multiplied by the number of processors online on the system, if that can be determined.

Sets the `algorithm` used for sorting the queue. Only the first character of the value is used. Legal values are:

- **host** Orders by the name of the first host name of the first recipient. Makes better use of the connection cache, but may tend to process low priority messages that go to a single host over high priority messages that go to several hosts; it probably should not be used on slow network links.

- **filename** Orders by the name of the queue file name. Saves the overhead of reading all of the queued items before starting the queue run.

- **time** Orders by the submission time. Should not be used because it allows large, bulk mail to go out before smaller, personal mail. May be appropriate on certain hosts with very fast connections.

- **priority** Orders by message priority. Is the default.

Do not use. Use `Timeout.queuereturn`.

Name of file containing random data or the name of the socket if EGD is used. A required prefix `egd:` or `file:` specifies the type. `STARTTLS` requires this filename if the compile flag `HASURANDOM` is not set (see `/user/samples/tcpip/sendmail/README`).

Set resolver options. Values can be set using `+flag` and cleared using `-flag`. Available flags are:

- `debug`
- `aonly`
- `usevc`
- `primary`
- `igtnc`
- `recurse`
- `defnames`
- `stayopen`
- `dnsrch`

The string `HasWildcardMX` (without a `+` or `-`) can be specified to turn off matching against MX records when doing name canonicalizations.

**Note:** In previous releases, this option indicated that the name server be responding in order to accept addresses. This has been replaced by checking to see if the DNS method is listed in the service switch entry for the hosts service.
RttImpliesDsn

If this option is set, a ReturnReceipt-To: header causes the request of a DSN to be sent to the envelope sender as required by RFC1891, not to the address given in the header.

RunAsUser=user

The user parameter may be a user name (looked up in /etc/passwd) or a numeric user ID. Either form can have :group attached, group can be numeric or symbolic. If set to a non-zero/non-root value, sendmail will change to this user ID shortly after startup. This avoids a certain class of security problems. However, this means that all .forward and :include: files must be readable by the indicated user and all files to be written must be writable by user. Also, all file and program deliveries will be marked unsafe unless the option DontBlameSendmail=NonRootAddrSafe is set, in which case the delivery will be done as user. It is also incompatible with the SafeFileEnvironment option. It may not actually add much to security on an average system, and may in fact detract from security, because other file permissions must be loosened. However, it may be useful on firewalls and other places where users do not have accounts and the aliases file is well constrained.

RecipientFactor= fact

The indicated factor is added to the priority for each recipient, thus lowering the priority of the job. This value penalizes jobs with large numbers of recipients. Defaults to 30000.

RefuseLA= LA

When the system load average exceeds LA, refuse incoming SMTP connections. Defaults to 12 multiplied by the number of processors online on the system, if that can be determined.

RetryFactor= fact

The factor is added to the priority every time a job is processed. Each time a job is processed, its priority will be decreased by the indicated value. In most environments this should be positive, because hosts that are down may be down for a long time. Default is 90000.

SafeFileEnvironment= dir

If this option is set, sendmail will do a chroot(2) call into the indicated directory before doing any file writes. If the file name specified by the user begins with dir, that partial path name will be stripped off before writing. For example, if the SafeFileEnvironment variable is set to /safe then aliases of /safe/logs/file and /logs/file actually indicate the same file. Additionally, if this option is set, sendmail will refuse to deliver to symbolic links.

SaveFromLine

Save From lines at the front of headers. They are assumed to be redundant and are discarded.

SendMimeErrors

If set, send error messages in MIME format (see RFC2045 and RFC1344 for details). If disabled, sendmail will not return the DSN keyword in response to an EHLO and will not do Delivery Status Notification processing as described in RFC1891.

ServerCertFile

File containing the certificate of the server. This certificate is used when sendmail acts as server.

ServerKeyFile

File containing the private key belonging to the server certificate.

ServiceSwitchFile= filename

If your host operating system has a service switch abstraction, that service will be consulted and this option is ignored. Otherwise, this is the name of a file that provides the list of methods used to implement particular services. The syntax is a series of lines, each of which is a sequence of words. The first word is the service name, and following words are service types. The services that sendmail consults directly are aliases and hosts. Service types can be dns, nis, nisplus, or files. The appropriate support must be compiled in before the service can be referenced. If ServiceSwitchFile is not specified, it defaults to /etc/mail/service.switch. If that file does not exist, the default switch is

```
aliases    files
hosts      dns nis files
```

The default file is /etc/mail/service.switch.

SevenBitInput

Strip input to seven bits for compatibility with old systems. This should not be necessary.
If set, **From**: lines that have embedded newlines are unwrapped onto one line. This is to get around a bug in Lotus Notes that apparently cannot understand legally wrapped RFC822 headers.

If set, a client machine will never try to open two SMTP connections to a single server machine at the same time, even in different processes. That is, if another sendmail is already talking to some host, a new sendmail will not open another connection. Although this reduces the load on the other machine, it can cause mail to be delayed. For example, if one sendmail is delivering a huge message, other sendmail processes will not be able to send even small messages. Also, it requires another file descriptor (for the lock file) per connection, so you may have to reduce the **ConnectionCacheSize** option to avoid running out of per-process file descriptors. Requires the **HostStatusDirectory** option.

Specifies the **message** to print when the SMTP server starts up. Defaults to $j Sendmail $v ready at $b.

Log summary statistics in the named **file**. If no file name is specified, statistics is used. If not set, no summary statistics will be saved. This file does not grow in size. It can be printed using the mailstats(8) program.

Always instantiate the queue file, even if you are going to attempt immediate delivery. Sendmail always instantiates the queue file before returning control to the client under any circumstances. This should always be set.

Specifies the file mode for queue files. It is interpreted in octal by default. Default is 0600.

Sets **time-out** values. For more information, see Read Timeouts on page 236.

Set the local time zone info to **tzinfo**. If this is not set, the TZ environment variable is cleared and the system default is used. If set but null, the user's TZ variable is used. If set and non-null, the TZ variable is set to this value.

The user parameter can be a user name (looked up in /etc/passwd) or a numeric user ID. Trusted user for file ownership and starting the daemon. If set, generated alias databases and the control socket (if configured) will automatically be owned by this user.

If this system is the best (that is, lowest preference) MX for a given host, its configuration rules should detect this situation and treat that condition specially by forwarding the mail to a UUCP feed, treating it as local, and so on. However, in some cases, such as in the case with Internet firewalls, you may want to try to connect directly to that host as though it had no MX records at all. Setting this option causes sendmail to try this. Unfortunately, errors in your configuration are likely to be diagnosed as ”host unknown” or ”message timed out” instead of something more meaningful. This option is not recommended.

Defines the format used when sendmail must add a UNIX-style **From** line, such as a line beginning From<space>user). Defaults to From $g $d. Do not change this unless your system uses a different mailbox format.

If set, .include: and .forward files that are group writable are considered unsafe, and they will not be able to reference programs or write directly to files. World writable .include: and .forward files are always unsafe.

The user database specification.

Run in Verbose mode. If this is set, sendmail adjusts options HoldExpensive and DeliveryMode so that all mail is delivered completely in a single job so that you can see the entire delivery process. The Verbose option should never be set in the configuration file; it is intended for command line use only.

Defines the **threshold** in bytes, before a memory-based queue transcript file becomes disk-based. The default is 4096 bytes.
All options can be specified on the command line using the -O or -o flag, but most will cause sendmail to relinquish its setuid permissions. The options that will not cause this are SevenBitInput, EightBitMode, MinFreeBlocks, CheckpointInterval, DeliveryMode, ErrorMode, IgnoreDots, SendMimeErrors, LogLevel, OldStyleHeaders, PrivacyOptions, SuperSafe, Verbose, QueueSortOrder, MinQueueAge, DefaultCharSet, DialDelay, NoRecipientAction, ColonOkInAddr, MaxQueueRunSize, SingleLineFromHeader, and AllowBogusHELO. Actually, PrivacyOptions given on the command line are added to those already specified in the sendmail.cf file and cannot be reset. Also, M (define macro) when defining the r or s macros is also considered safe.

**P - Precedence Definitions:** Values for the “Precedence:” field may be defined using the P control line. The syntax of this field is:

```
Pname=num
```

When the name is found in a “Precedence:” field, the message class is set to num. Higher numbers mean higher precedence. Numbers less than zero have the special property that if an error occurs during processing, the body of the message will not be returned; this is expected to be used for “bulk” mail such as through mailing lists. The default precedence is zero. For example, the list of default precedences is:

- Pfist-class=0
- Pspecial-delivery=100
- Plist=-30
- Pbulk=-60
- Pjunk=-100

**V - Configuration Version Level:** To provide compatibility with old configuration files, the V line has been added to define basic semantics of the configuration file. This is not intended as long term support. These compatibility features may be removed in future releases.

**Note:** Configuration version levels are independent of configuration file version numbers. For example, version number 8.9 configuration files use version level 8 configurations.

"Old" configuration files are defined as version level one.

Version level two files make the following changes:

1. Host name canonicalization ([$...$]) appends a dot if the name is recognized. This gives the configuration file a way to determine if a match occurred. This initializes the host map with the -a. flag. You can reset it to anything else by declaring the map explicitly.
2. Default host name extension is consistent throughout processing. Version level one configurations turned off domain extension during certain points in processing by adding the local domain name. Version level two configurations include a trailing dot to indicate that the name is already canonical.
3. Local names that are not aliases are passed through a new distinguished ruleset five. This can be used to append a local relay. This can be prevented by resolving the local name by using the @ symbol as a prefix (for example, @vikki). Something that resolves to a local mailer and a user name of vikki will be passed through ruleset five, but a user name of @vikki will have the @ prefix stripped, will not be passed through to ruleset five, but will otherwise be treated the same as the prior example. The exception is that this might be used to implement a policy where mail sent to vikki is handled by a central hub but mail sent to vikki@localhost is delivered directly.

Version level three files allow # initiated comments on all lines. Exceptions are backslash escaped # marks and the $# syntax.

Version level four files are equivalent to level three files.

Version level five files change the default definition of $w to be the first component of the hostname.
Version level six configuration files change many of the local processing options (i.e., aliasing and matching the address beginning for the | character) to mailer flags. This allows fine grained control over the special local processing. Version level six files may also use long option names. The ColonOkInAddr option (which allows colons in the local part of the address) defaults to on in configuration files with lower version numbers. The configuration file requires additional "intelligence" to properly handle the RFC 822 group construct.

Version level seven configuration files use new option names to replace old macros.

<table>
<thead>
<tr>
<th>Option</th>
<th>Old Macro</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e</td>
<td>SmtpGreetingMessage</td>
</tr>
<tr>
<td>$1</td>
<td>UnixFromLine</td>
</tr>
<tr>
<td>$o</td>
<td>OperatorChars</td>
</tr>
</tbody>
</table>

Prior to version seven, the F=q flag (use the return value 250 instead of 252 for SMTP VRFY commands) was assumed.

Version level eight configuration files allow $# on the left side of ruleset lines.

Version level nine configuration files allow parentheses in rulesets, which means they are not treated as comments and are removed.

The V line may have an optional /vendor variable to indicate that the configuration file uses vendor specific modifications. You may use /Berkeley to indicate that the file uses the Berkeley sendmail dialect.

**K - Key File Declaration:** Special maps can be defined using the line:

```
Kmapname mapclass arguments
```

The mapname is the name by which this map is referenced in the rewrite rules. The mapclass is the name of a type of map; these are compiled in to sendmail. The arguments are interpreted depending on the class; typically, there would be a single argument naming the file containing the map.

Maps are referenced using the syntax:

```
$( map key $@ arguments $: default $)
```

where either or both of the arguments or default portion may be omitted. The $@ arguments may appear more than once. The indicated key and arguments are passed to the appropriate mapping function. If it returns a value, it replaces the input. If it does not return a value and the default is specified, the default replaces the input. Otherwise, the input is unchanged.

During replacement of either a map value or default, the string "%n" (where n is a digit) is replaced by the corresponding argument. Argument zero is always the database key. For example, the rule:

```
R$- ! $+ $: $(uucp $1 $0 $2 $: %1 %0 . UUCP $)
```

looks up the UUCP name in a (user-defined) UUCP map. If not found, it turns it into ".UUCP" form. The database might contain records like:

```
devvax %10 %0.DEC.COM
research %10%0.ATT.COM
```

**Note:** The default clauses never perform this mapping.

The built-in map with both name and class "host" is the host name canonicalization lookup. Thus, the syntax:

```
$(host hostname$)
```
is equivalent to:

```
$[hostname$]
```

There are many defined classes.

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dbm</td>
<td>Database lookups using the ndbm(3) library. Sendmail must be compiled with NDBM defined.</td>
</tr>
<tr>
<td>btree</td>
<td>Database lookups using the btree interface to the Berkeley DB library. Sendmail must be compiled with NEWDB defined.</td>
</tr>
<tr>
<td>hash</td>
<td>Database lookups using the hash interface to the Berkeley DB library. Sendmail must be compiled with NEWDB defined.</td>
</tr>
<tr>
<td>nis</td>
<td>NIS lookups. Sendmail must be compiled with NEWDB defined.</td>
</tr>
<tr>
<td>nisplus</td>
<td>NIS+ lookups. Sendmail must be compiled with NISPLUS defined. The argument is the name of the table to use for lookups, and the -k and -v flags may be used to set the key and value columns respectively.</td>
</tr>
<tr>
<td>ldap</td>
<td>LDAP X500 directory lookups. Sendmail must be compiled with LDAPMAP defined. The map supports most of the standard arguments and command line arguments of the ldapsearch program. By default, if a single query matches multiple values, only the first value will be returned unless the -z (value separator) map flag is set. Also, the -1 map flag will treat a multiple value return as if there were no matches.</td>
</tr>
<tr>
<td>ldapx</td>
<td>LDAP X500 directory lookups. Sendmail must be compiled with LDAPMAP defined. The map supports most of the standard arguments and command line arguments of the ldapsearch program.</td>
</tr>
<tr>
<td>text</td>
<td>Text file lookups. The format of the text file is defined by the -k (key field number), -v (value field number), and -z (field delimiter) flags.</td>
</tr>
<tr>
<td>stab</td>
<td>Internal symbol table lookups. Used internally for aliasing.</td>
</tr>
<tr>
<td>implicit</td>
<td>Really should be called “alias.” This is used to get the default lookups for alias files, and is the default if no class is specified for alias files.</td>
</tr>
<tr>
<td>user</td>
<td>Looks up users using getpwnam(3). The -v flag can be used to specify the name of the field to return (although this is normally used only to check the existence of a user).</td>
</tr>
<tr>
<td>host</td>
<td>Canonicalizes host domain names. Given a host name, it calls the name server to find the canonical name for that host.</td>
</tr>
<tr>
<td>bestmx</td>
<td>Returns the best MX record for a host name given as the key. The current machine is always preferred. For example, if the current machine is one of the hosts listed as the lowest preference MX record, it will be guaranteed to be returned. This can be used to find out if this machine is the target for an MX record and mail can be accepted on that basis. If the -z flag is given, all MX names are returned (separated by the given delimiter).</td>
</tr>
<tr>
<td>sequence</td>
<td>The arguments on the “K” line are a list of maps; the resulting map searches the argument maps in order until it finds a match for the indicated key. For example, if the key definition is: Kmap1 ... Kmap1 ... Kseqmap sequence map1 map2 then a lookup against “seqmap” first does a lookup in map1. If that is found, it returns immediately. Otherwise, the same key is used for map2.</td>
</tr>
<tr>
<td>syslog</td>
<td>The key is logged via syslogd(8). The lookup returns the empty string.</td>
</tr>
<tr>
<td>switch</td>
<td>Much like the “sequence” map except that the order of maps is determined by the service switch. The argument is the name of the service to be looked up; the values from the service switch are appended to the map name to create new map names. For example, consider the key definition: Kali switch aliases together with the service switch switch entry: aliases nis files This causes a query against the map “ali” to search maps named “ali.nis” and “ali.files” in that order.</td>
</tr>
</tbody>
</table>
Class

dequote

Description
Strip double quotes ("') from a name. It does not strip backslashes, and will not strip quotes if the resulting string would contain unscannable syntax (that is, basic errors like unbalanced angle brackets; more sophisticated errors such as unknown hosts are not checked). The intent is for use when trying to accept mail from systems such as DECnet that routinely quote odd syntax such as:

"49ers::ubell"

A typical use is probably something like:

```
Kdequote dequote
...
R$-  $: $(dequote $1 $)
R$+  $: $>$3 $1 $2
```

Care must be taken to prevent unexpected results; for example,

```
"|someprogram < input > output"
```

will have quotes stripped, but the result is probably not what was intended. Fortunately, these cases are rare.

regex

The map definition on the K line contains a regular expression. Any key input is compared to that expression using the POSIX regular expressions routines `regcomp()`, `regerr()`, and `regexec()`. Refer to the documentation for those routines for more information about regular expression matching. No rewriting of the key is done if the -m flag is used. Without it, the key is discarded, or if -s is used, it is substituted by the substring matches, delimited by the $| or the string specified with the -d flag. The flags available for the map are:

- `n` not
- `f` case sensitive
- `b` basic regular expressions (default is extended)
- `s` substring match
- `d` set the delimiter used for -s
- `a` append string to key
- `m` match only, do not replace/discard value

The -s flag can include an optional parameter which can be used to select the substrings in the result of the lookup. For example, -s1,3,4.

program

The arguments on the K line are the path name to a program and any initial parameters to be passed. When the map is called, the key is added to the initial parameters and the program is invoked as the default user/group ID. The first line of standard output is returned as the value of the lookup. This has many potential security problems and terrible performance. It should be used only when absolutely necessary.

macro

Set or clear a macro value. To set a macro, pass the value as the first argument in the map lookup. To clear a macro, do not pass an argument in the map lookup. The map always returns the empty string. Examples of typical usage includes:

```
Kstorage macro
```

```
# set macro $<MyMacro to the ruleset match
R$+  $:$<storage $<MyMacro> $0 $1 $) $1
# set macro $<MyMacro to an empty string
R$+  $:$<storage $<MyMacro> $0 $) $1
# set macro $<MyMacro
R$-  $:$<storage $<MyMacro> $) $1
```
Class | Description
--- | ---
arith | Perform simple arithmetic operations. The operation is given as key, currently +, -, *, /, l (for less than), and <= are supported. The two operands are given as arguments. The lookup returns the result of the computation (True or False) for comparisons, integer values otherwise. All options that are possible for maps are ignored. A simple example is:
Kcomp arith
...
Scheck_etrn
R$*$: $(\text{comp l } @ \text{load_avg}) $0 7 $1
RFALSE $# error ...

Most of these accept as arguments the same optional flags and a filename (or a mapname for NIS; the filename is the root of the database path, so that .db or some other extension appropriate for the database type will be added to get the actual database name). Known flags are:

<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-o</td>
<td>Indicates that this map is optional. That is, if it cannot be opened, no error is produced, and sendmail will behave as if the map existed but was empty.</td>
</tr>
<tr>
<td>-N, -O</td>
<td>If neither -N or -O are specified, sendmail uses an adaptive algorithm to decide whether or not to look for null bytes on the end of keys. It starts by trying both; if it finds any key with a null byte, it never tries again without a null byte and vice versa. If -N is specified, it never tries without a null byte and if -O is specified, it never tries with a null byte. Setting one of these can speed matches but are never necessary. If both -N and -O are specified, sendmail will never try any matches at all. That is, everything will appear to fail.</td>
</tr>
<tr>
<td>-ax</td>
<td>Append the string x on successful matches. For example, the default host map appends a dot on successful matches.</td>
</tr>
<tr>
<td>-Tx</td>
<td>Append the string x on temporary failures. For example, x would be appended if a DNS lookup returned server failed or an NIS lookup could not locate a server. See the -t flag for additional information.</td>
</tr>
<tr>
<td>-f</td>
<td>Do not fold upper to lower case before looking up the key.</td>
</tr>
<tr>
<td>-m</td>
<td>Match only (without replacing the value). If you only care about the existence of a key and not the value (as you might when searching the NIS map &quot;hosts.byname&quot; for example), this flag prevents the map from substituting the value. However, The -a argument is still appended on a match, and the default is still taken if the match fails.</td>
</tr>
<tr>
<td>-kkeycol</td>
<td>The key column name (for NIS+) or number (for text lookups).</td>
</tr>
<tr>
<td>-valcol</td>
<td>The value column name (for NIS+) or number (for text lookups).</td>
</tr>
<tr>
<td>-zdelim</td>
<td>The column delimiter (for text lookups). It can be a single character or one of the special strings &quot;\n&quot; or &quot;\t&quot; to indicate newline or tab respectively. If omitted entirely, the column separator is any sequence of whitespace.</td>
</tr>
<tr>
<td>-t</td>
<td>Normally, when a map attempts to do a lookup and the server fails (e.g., sendmail could not contact any name server — this is not the same as an entry not being found in the map), the message being processed is queued for future processing. The -t flag turns off this behavior, letting the temporary failure (server down) act as though it were a permanent failure (entry not found). It is particularly useful for DNS lookups, where another’s misconfigured name server can cause problems on your machine. Care must be taken to avoid “bouncing” mail that would be resolved correctly if another attempt were made. A common strategy is to forward such mail to another mail server.</td>
</tr>
<tr>
<td>-D</td>
<td>Perform no lookup in deferred delivery mode. This flag is set by default for the host map.</td>
</tr>
<tr>
<td>-spacesub</td>
<td>The character to use to replace space characters after a successful map lookup. This is especially useful for regex and syslog maps.</td>
</tr>
<tr>
<td>-q</td>
<td>Do not dequote the key before lookup.</td>
</tr>
</tbody>
</table>
Flag | Description
--- | ---
-A | When rebuilding an alias file, the -A flag causes duplicate entries in the text version to be merged. For example, the following two entries:

```
list: user1,user2
list: user3
```

would be treated as if they were the following single entry:
```
list: user1,user2,user3
```

The following additional flags are present in the LDAP map only:

<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-R</td>
<td>Do not auto chase referrals. Sendmail must be compiled with <code>-DLAP_REFERRALS</code> to use this flag.</td>
</tr>
<tr>
<td>-n</td>
<td>Retrieve attribute names only.</td>
</tr>
<tr>
<td>-rderf</td>
<td>Set the alias dereference option to one of the following: <code>never</code>, <code>always</code>, <code>search</code>, or <code>find</code>.</td>
</tr>
<tr>
<td>-sscope</td>
<td>Set search scope to one of the following: <code>base</code>, <code>one</code> (one level), or <code>sub</code> (subtree).</td>
</tr>
<tr>
<td>-host</td>
<td>LDAP server host name.</td>
</tr>
<tr>
<td>-base</td>
<td>LDAP search base.</td>
</tr>
<tr>
<td>-pport</td>
<td>LDAP service port.</td>
</tr>
<tr>
<td>-timelimit</td>
<td>Time limit for LDAP queries.</td>
</tr>
<tr>
<td>-sizelimit</td>
<td>Size (number of matches) limit for LDAP queries.</td>
</tr>
<tr>
<td>-distinguished_name</td>
<td>The distinguished name to use to log in to the LDAP server.</td>
</tr>
<tr>
<td>-method</td>
<td>The method to authenticate to the LDAP server. Should be one of the following: <code>LDAP_AUTH_NONE</code>, <code>LDAP_AUTH_SIMPLE</code>, or <code>LDAP_AUTH_KRBV4</code>.</td>
</tr>
<tr>
<td>-passwordfile</td>
<td>The file containing the secret key for the <code>LDAP_AUTH_SIMPLE</code> authentication method or the name of the Kerberos ticket file for <code>LDAP_AUTH_KRBV4</code>.</td>
</tr>
<tr>
<td>-1</td>
<td>Force LDAP searches to succeed only if a single match is found. If multiple values are found, the search will be treated as if no match was found.</td>
</tr>
</tbody>
</table>

The dbm map appends the strings `.pag` and `.dir` to the given filename; the two db-based maps append `.db`. For example, the map specification

```
Kuucp dbm -o -N /usr/lib/uucpmap
```

specifies an optional map named "uucp" of class "dbm"; it always has null bytes at the end of every string, and the data is located in `/usr/lib/uucpmap.(dir,pag)`.

**Commands and Operands**

**Command and Operand** | **Description**
--- | ---
CXWord1 Word2... | Defines the class of words that can be used to match the left-hand side of rewrite rules. Class specifiers (X) may be any of the uppercase letters from the ASCII character set. Lowercase letters and special characters are reserved for system use.
 DXValue | Defines a macro (X) and its associated Value. Macro specifiers may be any of the uppercase letters from the ASCII character set. Lowercase letters and special characters are reserved for system use.
 FXFileName [Format] | Reads the elements of the class (X) from the FileName variable, using an optional scanf format specifier. The format specifier contains only one conversion specification. One class number is read for each line in the FileName variable.

Chapter 1. System Files 267
## Command and Operand

### H(?MFlags?)HeaderName:
**Description**
Defines the header format the `sendmail` command inserts into a message. Continuation lines are a part of the definition. The `HeaderTemplate` is macro-expanded before insertion into the message. If the `MFlags` are specified and at least one of the specified flags is included in the mailer definition, this header is automatically written to the output message. If the header appears in the input message, it is written to the output message regardless of the `MFlags` variable.

### MName,[Field=Value]
**Description**
Defines a Mail program where the `Name` variable is the name of the Mail program and `Field=Value` pair defines the attributes of the mailer.

### Ox[Value]
**Description**
Sets the option to the value of `x`. If the option is a valued option, you must also specify the `Value` variable. Options may also be selected from the command line. **Note:** For valid values, see “O — Set Option” on page 247.

### PName=Number
**Description**
Defines values for the `Precedence:` header field. When the `Name` variables found in a message's `Precedence:` field, the message's precedence is set to the `Number` variable. Higher numbers indicate higher precedences. Negative numbers indicate that error messages are not returned. The default `Number` is 0.

### RLeftHandSide RightHandSide Comments
**Description**
Defines a rewrite rule. One or more tab characters separate the three fields of this command. If space characters are used as field separators, option `J` must be set. The `J` option allows spaces as well as tabs to separate the left- and right-hand sides of rewrite rules. The `J` option allows rewrite rules to be modified using an editor that replaces tabs with spaces.

### Sx
**Description**
Sets the rule set currently defined to the specified number(`x`). If a rule set definition is started more than once, the new definition overwrites the old.

### TUser1 User2 ...
**Description**
Defines user IDs for the system administrators. These IDs have permission to override the sender address using the `-f` flag. More than one ID can be specified per line.

## Files

- `/etc/mail/sendmail.cf`
  Specifies the path of the `sendmail.cf` file.
- `/etc/passwd`
  Contains basic user attributes.
- `/etc/mail/aliases`
  Contains alias definitions for the `sendmail` command.

## Related Information

The `sendmail` command.

The `/etc/passwd` file.

## setinfo File

### Purpose
Describes the format of a set characteristics file.

### Description
The `setinfo` file is an ASCII file that describes the characteristics of the set along with information that helps control the flow of installation. It is created by the software set developer and is included in the Set Installation Package (SIP). A SIP is a special purpose package that controls the installation and removal of a set of packages.

Each entry in the `setinfo` file is a line that consists of predefined fields. Each entry corresponds to a package belonging to the set and must contain the following `<tab>`-separated fields:

1. **Package Abbr**
   This field contains the abbreviated name of the package. The abbreviation must be a short string
(no more than nine characters long) and must conform to the file naming rules. All characters in the abbreviation must be alphanumeric and the first character cannot be numeric. install, new, and all are reserved.

This abbreviated name must be the same as the one used in pkginfo.

2. **Parts**
   This field specifies the number of parts this package consists of.

3. **Default**
   This field contains the character 'y' indicating that the package is to be installed as a default. Conversely, an 'n' indicates that the package will not be installed.

4. **Category**
   The category under which the package belongs. Release 4 defines four categories: "application," "graphics," "system" and "utilities." All packages must be assigned to one of these categories. If you choose, you can also assign a package to a category you defined. Categories are case-insensitive and may contain only alphanumerics. Each category is limited to 16 characters.

5. **Package Full-Name**
   Text that specifies the package name (maximum length of 256 ASCII characters). This field must be the same as NAME in the pkginfo file.

The order of the packages listed in the setinfo file must reflect any package dependencies (if any) and must represent the order in which packages occur on the media (in the case of datastream). Any package for which there exists a dependency must be listed prior to the package(s) that depends on it.

**Examples**

Shown below is a setinfo file for set admin:

```bash
# Format for the setinfo file. Field separator is: <tab>
# pkg parts default category pkg full-name
# abbr y/n

oam 4 y application OAM
bkrs 1 y system Extended Backup and Restore
face 1 y application FACE
```

**Related Information**
The pkginfo file format.

**setup.csh File**

**Purpose**
Sets the C-shell environment variables needed to build an InfoCrafter database.

**Description**
The setup.csh file defines C-shell environment variables necessary to build an InfoCrafter database from the command line. The setup.csh file contains the definition of the TOOLS_DIRS and TOLEVEL_BUILDDIR variables; if there are relative path names of source files in your input list, it also sets the TOLEVEL_SOURCEDIR variable. The TOOLS_DIRS variable is added to your path environment variable so you can use the icft command without specifying the full path name.

The default value for the TOOLS_DIRS environment variable is /usr/lpp/icraft/bin. The TOLEVEL_SOURCEDIR and TOLEVEL_BUILDDIR variables have no default values.
You must copy the **setup.csh** file from `/usr/lpp/icraft/bin` to another location (such as your home directory) and edit it to define the variables. Then, use the `source setup.csh` command to assign the new definitions to the variables.

**Examples**

A sample **setup.csh** file appears as follows:

```
setenv TOLEVEL_SOURCEDIR $HOME/desktop
setenv TOOLSDIR /usr/lpp/icraft/bin
setenv TOLEVEL_BUILDDIR $TOOLSDIR/master
```

To set the C-shell environment variables, enter the following:

```
source setup.csh
```

The following message is displayed:

```
setup.csh: assigning environment variables
for InfoCrafter...
```

**Files**

/`usr/lpp/icraft/bin/setup.csh`

Contains the definitions of C-shell environment variables.

**Related Information**

The **setup.sh** file.

**setup.sh** File

**Purpose**

Defines the Bourne or Korn shell environment variables needed to build an InfoCrafter database.

**Description**

The **setup.sh** file defines Bourne or Korn shell environment variables necessary to build an InfoCrafter database from the command line using the `icft` command. The **setup.sh** file sets the `TOOLSDIR` and `TOLEVEL_BUILDDIR` variables. If there are relative path names of source files in your input list, it also sets the `TOLEVEL_SOURCEDIR` variable. The `TOOLSDIR` variable is added to your path environment variable so you can enter the `icft` command without specifying the full path name.

Default value for the `TOOLSDIR` environment path variable is `/usr/lpp/icraft/bin`. `TOLEVEL_SOURCEDIR` and `TOLEVEL_BUILDDIR` have no default values.

You must copy the **setup.sh** file from `/usr/lpp/icraft/bin` to another location (such as your home directory) and edit it to define the variables. Then, use the `. setup.sh` command to set the variables to the defined values.

**Examples**

A sample **setup.sh** file appears as follows:

```
TOLEVEL_SOURCEDIR = $HOME/desktop
TOOLSDIR = /usr/lpp/icraft/bin
TOLEVEL_BUILDDIR = $TOOLSDIR/master
```

To set Bourne or Korn shell environment variables, enter the following:

```
. setup.sh
```
The following message is given:
setup.sh: assigning environment variables for InfoCrafter

Files

/usr/lpp/icraft/bin/setup.sh Contains definitions for Bourne and Korn shell environment variables.

Related Information

The setup.csh file.

slp.conf File

Purpose

File used by SLP APIs.

Description

To use the /etc/slp.conf file, ensure that all parameters for an entry are contained on one line in the configuration file. Comments can be included in the file beginning with a pound sign (#) in column 1.

The format is:

# <keyword> = <value>

or

# <keyword> = <value1>,<value2>,..., <valueN>

If there are more than 1 value to be configured, use a comma (,) as a separator.

If there are multiple lines configured against the same <keyword>, the first valid configured line is read, and the rest are ignored.

Parameters

net.slp.maxResults A 32-bit integer that gives the maximum number of results to accumulate and return for a synchronous request before the timeout. Positive integers and -1 are legal values. A value of -1 indicates that all results should be returned. The default value is -1. For example:

net.slp.maxResults = 35

net.slp.useScopes A value-list of strings that indicate the only scopes a UA or SA is allowed to use when making requests or registering. Otherwise, indicates the scopes a DA must support. The default scope "DEFAULT" is used if no other information is available. For example:

net.slp.useScopes = david,bob

net.slp.DAAddress A value-list of IP addresses or DNS resolvable host names giving the SLPv2 DAs to use for statically configured UAs and SAs. The default is none. For example:

net.slp.DAAddress = 9.3.149.20, blahblah.ibm.com
**net.slp.isBroadcastOnly**
A boolean indicating whether broadcast should be used instead of multicast. The default is false (that is, multicast is used). For example:

```c
net.slp.isBroadcastOnly = false
```

**net.slp.multicastTTL**
A positive integer less than or equal to 255, giving the multicast TTL. The default is 255 (in seconds). For example:

```c
net.slp.multicastTTL = 255
```

**net.slp.DAActiveDiscoveryInterval**
# A 16-bit positive integer giving the number of seconds between DA active discovery queries. If this parameter is set to 0, the active discovery is turned off. This property corresponds to the protocol specification parameter `CONFIG_DA_FIND`. The default is 900 (in seconds). For example:

```c
net.slp.DAActiveDiscoveryInterval = 1200
```

**net.slp.multicastMaximumWait**
A 32-bit integer giving the maximum amount of time to perform multicast, in milliseconds. This property corresponds to the `CONFIG_MC_MAX` parameter in the protocol specification. The default is 15000 (in ms). For example:

```c
net.slp.multicastMaximumWait = 10000
```

**net.slp.multicastTimeouts**
A value-list of 32-bit integers used as timeouts, in milliseconds, to implement the multicast convergence algorithm. Each value specifies the time to wait before sending the next request. This property corresponds to the `CONFIG_MC_RETRY` parameter in the protocol specification. The default is 3000,3000,3000,3000,3000 (in ms). There is no limitation on the maximum number entries to be specified. However the total sum of the entries cannot be greater than `CONFIG_MC_RETRY` (15000 in ms). For example:

```c
net.slp.multicastTimeouts = 2000, 3000, 4000
```

**net.slp.DADiscoveryTimeouts**
A value-list of 32-bit integers used as timeouts, in milliseconds, to implement the multicast convergence algorithm during active DA discovery. Each value specifies the time to wait before sending the next request. This property corresponds to the protocol specification parameter `CONFIG_RETRY`. The default is 2000,2000,2000,2000,3000,4000 (in ms). There is no limitation on the maximum number entries to be specified. However the total sum of the entries cannot be greater than `CONFIG_RETRY` (15000 in ms). For example:

```c
net.slp.DADiscoveryTimeouts = 2000, 3000, 4000
```

**net.slp.datagramTimeouts**
A value-list of 32-bit integers used as timeouts, in milliseconds, to implement unicast datagram transmission to DAs. The \( n \)th value gives the time to block waiting for a reply on the \( n \)th try to contact the DA. The sum of these values is the protocol specification property `CONFIG_RETRY_MAX`. The default is 2000, 2000, 2000, 2000, 3000, 4000 (in ms). There is no limitation on the maximum number of entries to be specified. However, the total sum of the entries cannot be greater than `CONFIG_RETRY_MAX` (15000 in ms). For example:

```c
net.slp.datagramTimeouts = 2000, 3000, 4000
```

### Related Information
The `SLPAttrCallback`, `SLPClose`, `SLPEscape`, `SLPFindAttrs`, `SLPFindScopes`, `SLPFindSvs`, `SLPFindSrvTypes`, `SLPFree`, `SLPGetProperty`, `SLPOpen`, `SLPParseSrvURL`, `SLPSrvTypeCallback`, `SLPSrvURLCallback`, and `SLPUneescape` subroutines.
smi.my File

Purpose
Provides sample SMI input to the mosy command.

Description
The /usr/samples/snmpd/smi.my file is a sample input file to the mosy command, which creates an objects definition file for use by the snmpinfo command. The mosy compiler requires its input file to contain the ASN.1 definitions described in the Structure and Identification of Management Information (SMI) RFC 1155 and the Management Information Base (MIB) RFC 1213. The smi.my file contains the syntax descriptions from the SMI RFC 1155.

The smi.my file begins with a definition of the SNMP subtree of the MIB as assigned by the Internet Activities Board (IAB). It then contains the syntax definitions defined in RFC 1155.

Comments are specified by - - (two dashes). A comment can begin at any location and extends to the end of the line.

The smi.my file was created by extracting the definitions from Chapter 6 of RFC 1155. This file is shipped as /usr/samples/snmpd/smi.my. The file is part of Simple Network Management Protocol Agent Applications in Network Support Facilities.

Files

<table>
<thead>
<tr>
<th>File Path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/usr/samples/snmpd/mibII.my</td>
<td>Contains the ASN.1 definitions for the MIB II variables defined in RFC 1213.</td>
</tr>
<tr>
<td>/etc/mib.defs</td>
<td>Defines the Management Information Base (MIB) variables the snmpd agent should recognize and handle. This file is in the format that the snmpinfo command requires.</td>
</tr>
</tbody>
</table>

Related Information

The mosy command, snmpinfo command.

The mibII.my file.

RFC 1155, RFC 1213.


smitacl.group File

Purpose
Contains the group access control list (ACL) definitions for the System Management Interface Tool (SMIT).

Description
The /etc/security/smitacl.group file contains the group ACL definitions for SMIT. This is an ASCII file that contains a stanza for each system group. Each stanza is identified by a group name followed by a : (colon) and contains attributes in the form Attribute=Value. Each attribute pair ends with a newline character as does each stanza.
The file supports a default stanza. If an attribute is not defined, either the default stanza or the default value for the attribute is used.

A stanza contains the following attribute:

```
screens                Describes the list of SMIT screens for this group. (It is of the type SEC_LIST.) Examples include:
    screens = *          # Permit all screen access.
    screens = !*         # Deny all screen access.
    screens =            # Allows no specific screens
    # (screens can be added on a per user basis)
    screens = user,group,!tcpip # Allow user & group
    # screens, but not
    # tcpip screen
```

For a typical stanza, see the "Examples" section. This file may viewed with the `lssec` command and modified with the `chsec` command.

The screen names specified in the screens attribute are SMIT fastpath values. Many SMIT fastpath values can be found in the commands documentation. The `smit` command may also be used to determine the fastpath of the current screen. Please see the `smit` command for more information.

**Security**

Access Control: This file grants read and write access to the root user, and read access to members of the security group.

**Examples**

1. A typical stanza looks like the following example for the group called `group`:
   ```
   group:
   screens = *
   ```

2. To allow the `mksysb` screen only for a member of a group called `bkupgrp`, remove the default access to all screens and specify only the `mksysb` screen for the `bkupgrp` group by typing:
   ```
   default:
   screens =
   bkupgrp:
   screens = mksysb
   ```

   **Note:** Changing the default stanza will remove the default access to all screens for members of other groups.

3. To allow members of the `security` group to view the screens for user and group administration, add the following stanza:
   ```
   security:
   screens = top_menu,security,users,groups
   ```

   Additional screens may be granted to individual users by adding stanzas for each user to the `smitacl.user` file.

**Files**

/`etc/security/roles` Contains the list of valid roles.
/`etc/security/user.roles` Contains the list of roles for each user.
/`etc/security/smitacl.group` Contains the group ACL definitions.
/`etc/security/smitacl.user` Contains the user ACL definitions.
Related Information
The `smit`, `chsec`, and `lssec` commands.
The `getusraclattr` subroutine, `nextusracl` subroutine, `putusraclattr` subroutine.

smitacl.user File

Purpose
Contains the user access control list (ACL) definitions for the System Management Interface Tool (SMIT).

Description
The `/etc/security/smitacl.user` file contains the ACL definitions for SMIT. This is an ASCII file that contains a stanza for each system user. Each stanza is identified by a user name followed by a : (colon) and contains attributes in the form `Attribute=Value`. Each attribute pair ends with a newline character as does each stanza.

The file supports a default stanza. If an attribute is not defined, either the default stanza or the default value for the attribute is used.

A stanza contains the following attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>screens</td>
<td>Describes the list of SMIT screens for the user. (It is of the type <code>SEC_LIST</code>.) Examples include:</td>
</tr>
<tr>
<td></td>
<td><code>screens = *</code> # Permit all screen access.</td>
</tr>
<tr>
<td></td>
<td><code>screens = !*</code> # Deny all screen access.</td>
</tr>
<tr>
<td></td>
<td><code>screens =</code> # Allows no specific screens</td>
</tr>
<tr>
<td></td>
<td><code>(screens can be added on a per user basis)</code></td>
</tr>
<tr>
<td></td>
<td><code>screens = user,group,!tcpip</code> # Allow user &amp; group</td>
</tr>
<tr>
<td></td>
<td><code>screens, but not</code></td>
</tr>
<tr>
<td></td>
<td><code># tcpip screen</code></td>
</tr>
<tr>
<td>funcmode</td>
<td>Describes if the role database and/or SMIT ACL database should be used to determine accessibility. It also describes how to combine the <code>screens</code> data from the two databases. (It is of the type <code>SEC_CHAR</code>.) Examples include:</td>
</tr>
<tr>
<td></td>
<td><code>funcmode = roles+acl</code> # Use both roles and SMIT ACL # databases.</td>
</tr>
<tr>
<td></td>
<td><code>funcmode = roles</code> # Use only the roles database.</td>
</tr>
<tr>
<td></td>
<td><code>funcmode = acl</code> # Use only the SMIT ACL # database.</td>
</tr>
</tbody>
</table>

The defined values for `funcmode` are:
- `roles` Only the screen values from the roles database are used.
- `acl` Only the screen values from the SMIT ACL database are used.
- `roles+acl` The screen values from both the roles and the SMIT ACL databases are used.

For a typical stanza, see the `Examples` section. This file may viewed with the `lssec` command and modified with the `chsec` command.

The screen names specified in the `screens` attribute are SMIT fastpath values. Many SMIT fastpath values can be found in the commands documentation. The `smit` command may also be used to determine the fastpath of the current screen. Please see the `smit` command for more information.

Security
Access Control: This file grants read and write access to the root user, and read access to members of the security group.
Examples

1. To allow `pwduser` to access the groups which are accessible via the `users` and `passwd` SMIT fastpaths, type the following:

   ```
   pwduser:
     funcmode = roles+acl
     screens = users,passwd
   ```

2. To allow the `mksysb` screen only for user `bkupuser`, add the following stanza:

   ```
   bkupuser:
     screens = mksysb
   ```

Files

- `/etc/security/roles`: Contains the list of valid roles.
- `/etc/security/user.roles`: Contains the list of roles for each user.
- `/etc/security/smitacl.group`: Contains the group ACL definitions.
- `/etc/security/smitacl.user`: Contains the user ACL definitions.

Related Information

The `smit`, `chsec` and `lssec` commands.

The `getusraclattr` subroutine, `nextusracl` subroutine, `putusraclattr` subroutine.

**snmpd.conf File**

**Purpose**

Defines a sample configuration file for the `snmpdv1` agent.

**Description**

The `snmpd.conf` file provides the configuration information for the `snmpdv1` agent. This file can be changed while the `snmpdv1` agent is running. If the `refresh` or `kill -1` command is issued, the `snmpdv1` agent will reread this configuration file. The `snmpdv1` agent must be under System Resource Control (SRC) for the `refresh` command to force the reread. This file is part of Simple Network Management Protocol Agent Applications in Network Support Facilities.

This configuration file contains:

- Entries for Community names
- Access privileges and view definitions for incoming Simple Network Management Protocol (SNMP) request packets
- Entries for host destinations for trap notification
- Entries for log file characteristics
- Entries for `snmpd-specific parameters`
- Entries for SNMP Multiplexing Protocol (SMUX) association configurations
- Entries for the `sysLocation` and `sysContact` variables.

The `snmpd.conf` file must be owned by the root user. If the `snmpd.conf` file is not owned by root, or if the `snmpdv1` daemon cannot open the configuration file, the `snmpdv1` daemon issues a FATAL message to the log file if logging is enabled and `snmpdv1` terminates.

Certain rules apply for specifying particular parameters in entries in the `snmpd.conf` configuration file. Some entries require the specification of object identifiers or object names or both. The following rules apply:
1. An object identifier is specified in dotted numeric notation and must consist of at least three elements. The maximum number of elements in the object identifier is 50. Elements are separated by a . (dot). The first element must be a single digit in the range of 0 to 2. The second element must be an integer in the range of 1 to 40. The third and subsequent elements must be integers in the range of 1 to the size of an unsigned integer.

2. An object name consists of a textual name with an optional numeric instance. The object name must be known to the `snmpdv1` agent. Object names typically are names of nodes in the Management Information Base (MIB) tree. If the root of the MIB tree, iso, is specified as an object name, the numeric instance is absolutely required. A . (dot) separates the textual name from the numeric instance.

Community Entry
The community entry specifies the communities, associated access privileges and MIB views the `snmpdv1` agent allows. See example 1 for a sample entry. A community entry must be in the following format:

```
community CommunityName IPAddress NetMask Permissions ViewName
```

The following definitions apply to the variables in a community entry:

- **CommunityName**
  - The community name.

- **IPAddress**
  - The host name or IP address in dotted-decimal format for the specified community name.

- **NetMask**
  - A network mask in dotted-decimal format for the specified hostname or IP address.

- **Permissions**
  - Specifies one of:
    - `readOnly`
    - `writeOnly`
    - `readWrite`
    - `none`.

  The `Permissions` string is case-insensitive.

- **ViewName**
  - A unique object identifier in dotted numeric notation that is associated with a portion of the MIB tree to which the specified community name allows access. The `ViewName` value is the same as that specified in the view entry.

The minimum specification required for a community entry is:

```
community CommunityName
```

The default values for this minimum community entry are:

- **IPAddress**
  - 0.0.0.0

- **NetMask**
  - 0.0.0.0

- **Permissions**
  - `readOnly`

- **View**
  - iso.3

Fields to the right of the minimum entry are optional, with the limitation that no fields to the left of a specified field are omitted. Any information to the right of the `ViewName` variable is ignored. If an `IPAddress` of 0.0.0.0 is specified, the default `NetMask` is 0.0.0.0. If an `IPAddress` other than 0.0.0.0 is specified, the default `NetMask` is 255.255.255.255.

The `Permissions` default is `readOnly`. If the `ViewName` is not specified, the view for this community defaults to ISO, the entire MIB tree. For example:

```
community public 192.100.154.1
```
is a valid entry with the default values:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>NetMask</td>
<td>255.255.255.255</td>
</tr>
<tr>
<td>Permissions</td>
<td>readOnly</td>
</tr>
<tr>
<td>View</td>
<td>iso.3</td>
</tr>
</tbody>
</table>

The following entry is not valid because the required NetMask variable to the left of the Permissions variable is not specified:

```
community public  192.100.154.1
readWrite
```

In this case, the value in the Permissions variable is accepted as the NetMask value. Since the value in the Permissions variable is not in the format required for the NetMask variable, an error will occur. The snmpdv1 agent logs an EXCEPTIONS message if logging is enabled. In the case of an invalid community entry, the snmpdv1 agent ignores the entry.

**View Entry**

The view entry specifies the MIB subtrees to which a particular community has access. See example 3 for a sample entry. A view entry must be in the following format:

```
view ViewName  MibSubtree...
```

The following definitions apply to the variables in the view entry:

- **ViewName**: Specifies a unique object identifier in dotted-numeric notation that is associated with a portion of the MIB tree. This ViewName value is the same as that in the community entry and must be formatted as described there.

- **MibSubtree**: A list of MIB subtrees, or MIB groups, specified as either an object name or an object identifier, that is associated with the ViewName variable. If the MIBSubtree list is not specified, the view defaults to iso, the entire MIB tree.

Together, the view entry and its associated community entry define an access privilege or MIB view allowed by the snmpdv1 agent.

In the case of an invalid view entry, the snmpdv1 agent logs an EXCEPTIONS message, if logging is enabled, and ignores the view entry.

If a ViewName is specified in the community entry, but there is no view entry to describe that ViewName, snmpdv1 agent logs an EXCEPTIONS message stating that there is no such view for the community. The snmpdv1 agent will allow no access for that community and view association.

**Trap Entry**

The trap entry specifies the hosts the snmpdv1 agent notifies in the event a trap is generated. See Example 2 for a sample entry. A trap entry must be in the following format:

```
trap CommunityName  IPAddress  ViewName  TrapMask
```

In this format, the variable definitions are as follows:

- **CommunityName**: The community name to be encoded in the SNMP trap packet.
- **IPAddress**: The host name or IP Address in dotted-decimal format for the specified CommunityName.
- **ViewName**: The snmpdv1 agent only checks the ViewName to verify that the format is valid and that there are no duplicate ViewName variables specified.
The trap mask in hexadecimal format. The bits from left to right stand for coldStart trap, warmStart trap, linkDown trap, linkUp trap, authenticationFailure trap, egpNeighborLoss trap, and enterpriseSpecific trap. The rightmost bit does not have any meaning. A value of 1 will enable the corresponding trap to be sent. Otherwise, the trap is blocked.

For example:

<table>
<thead>
<tr>
<th>hexadecimal</th>
<th>bits</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>fe</td>
<td>1111 1110</td>
<td>block no traps</td>
</tr>
<tr>
<td>7e</td>
<td>0111 1110</td>
<td>block coldStart trap</td>
</tr>
<tr>
<td>be</td>
<td>1011 1110</td>
<td>block warmStart trap</td>
</tr>
<tr>
<td>3e</td>
<td>0011 1110</td>
<td>block coldStart trap and warmStart trap</td>
</tr>
</tbody>
</table>

The minimum specification required for a trap entry is:

```
trap CommunityName IPAddress
```

The default value of TrapMask for this minimum trap entry is fe. There is no trap blocked for this case.

Fields to the right of the minimum entry are optional, with the limitation that no fields to the left of a specified field are omitted. There should be no information to the right of the TrapMask variable.

In the case of an invalid trap entry, the snmpdv1 agent places an EXCEPTIONS message in the log file if logging is enabled and ignores the trap entry.

It is assumed that all hosts listed in the trap entries are listening on well-known UDP port 162 for SNMP traps. Because community views for traps are not supported, the snmpdv1 agent will send trap messages for all traps generated as indicated by the TrapMask variable to the hosts listed in the trap entries. If no trap entry appears in the snmpd.conf file, the snmpdv1 agent will not send out trap messages upon the generation of a trap.

Logging Entry
The logging entry specifies the characteristics for the snmpdv1 agent logging activities if logging is not directed from the snmpd command with the -f option. See example 4 for a sample entry. A logging entry must be in the following format:

```
logging FileName Enablement
logging size=Limit level=DebugLevel
```

The following definitions apply to the fields in the logging entries:

- **FileName** Specifies the complete path and file name of the log file.
- **Limit** Specifies the maximum size in bytes of the specified log file. If the limit is specified as 0, the file size is unlimited.
- **DebugLevel** Specifies the level of logging, which can be one of the following:
  - 0: All NOTICES, EXCEPTIONS, and FATAL messages
  - 1: Level 0 plus DEBUG messages
  - 2: Level 1 plus a hexadecimal dump of incoming and outgoing packets
  - 3: Level 2 plus an English version of the request and response packets
- **Enablement** Specifies whether logging is active. The following options are available:
  - **enabled**: Turns logging on.
  - **disabled**: Turns logging off.
There is no default log file. The *Enablement* default is disabled. The log file size *Limit* default is 0, which means unlimited. The *DebugLevel* default is 0 if the *snmpd* command is invoked without the `-d` option. If the `-d` option is specified, the default *DebugLevel* is the value specified by the `-d` option on the *snmpd* command line.

The `size=` and `level=` entries are absolutely required if a size or debug level are specified. There can be no spaces around the `=` (equal sign).

There are no restrictions regarding the order in which the variables are entered in the logging entries. A logging entry can contain single or multiple variables.

If the value for the `size=` field or *DebugLevel* variable cannot be converted into an integer, the default size and debug level are used. Because the *snmpd* command sets the log file configuration parameters immediately upon reading them, the parameters in the logging entry are not necessarily ignored if the *snmpd* command determines there is an invalid field in that entry. For example, in the following invalid logging entry:

```
logging size=100000 garbagestuff enabled
```

The *snmpd* command will set the size parameter, but will discard all information from the field value of `garbagestuff` to the end of the line. In addition, an EXCEPTIONS message will be logged if logging is enabled.

**snmpd Entry**

The *snmpd* entry specifies configuration parameters for the *snmpdv1* agent. See example 5 for a sample entry. An *snmpd* entry must be in the following format:

```
  snmpd  Variable=Value
```

The `=` (equal sign) is absolutely required; there can be no spaces around it.

The following definitions apply to the *snmpd* entry:

- **Variable** Specifies the specific configuration parameter. *Variable* can be one of the following values:
  - `maxpacket`
  - `querytimeout`.
- **Value** Specifies the value of the specific variable.

The configurable variables and allowable values are:

- **maxpacket**
  - The maximum packet size, in bytes, that the *snmpdv1* agent will transmit. The minimum to which this variable can be set is 300 bytes. The maximum value to which this variable can be set is 56KB. If there is no *snmpd* entry for `maxpacket`, the system socket default levels will be used.

- **querytimeout**
  - The time interval in seconds at which the *snmpdv1* agent will query the interfaces to check for interface status changes. The minimum value to which `querytimeout` can be set is 30 seconds. If 0 is specified, the *snmpdv1* agent will not query the interfaces for status changes. If there is no *snmpd* entry for `querytimeout`, the default value of 60 seconds is used.

The `=` (equal sign) is absolutely required; there can be no white space around it. There are no restrictions on the order in which the variables are entered in the *snmpd* entry. An *snmpd* entry can contain single or multiple variables.
The **snmpdv1** agent sets the **snmpd** specific parameters immediately upon reading them. If the values are invalid, the **snmpdv1** agent ignores them. If the **snmpdv1** agent encounters an invalid field in the entry, processing is terminated for that entry and the **snmpdv1** agent logs an EXCEPTIONS message if logging is enabled.

**smux Entry**

The smux entry specifies configuration information for SMUX associations between the **snmpdv1** agent and SMUX peer clients. See example 6 for a sample entry. A smux entry must be in the following format:

```
smux  ClientOID  Password  IPAddress  NetMask
```

The following definitions apply to the **smux** entry:

- **ClientOID**: Specifies the unique object identifier in dotted numeric notation of the SMUX peer client. The `ClientOID` must match the `ObjectID` specified in the `/etc/snmpd.peers` file.

- **Password**: Specifies the password that the **snmpdv1** agent requires from the SMUX peer client to authenticate the SMUX association. The `Password` must match the `Password` in the `/etc/snmpd.peers` file.

- **IPAddress**: The hostname or IP address in dotted notation of the host on which the SMUX peer client is executing.

- **NetMask**: Specifies a network mask in dotted decimal notation for the specified hostname or IP address.

The minimum specification for the **smux** entry is:

```
smux  ClientOID  Password
```

The default values for this minimum smux entry are:

- **IPAddress**: 127.0.0.1
- **NetMask**: 255.255.255.255

Fields to the right of the minimum entry are optional, with the limitation that no fields to the left of a specified field are omitted. Any information to the right of `NetMask` is ignored. If no password is specified, there is no confirmation for the SMUX association. If neither the `IPAddress` nor `NetMask` are specified, the SMUX association is limited to the local host.

In the case of an invalid smux entry, the **snmpdv1** agent logs an EXCEPTIONS message if logging is enabled and the **snmpdv1** agent ignores that smux entry.

**sysLocation and sysContact Entry**

The `sysLocation` and `sysContact` entries specify the values of the `sysLocation` and `sysContact` variables. The entry is specified in the following format:

```
sysLocation  "Austin, Texas, USA, XYZ, Bld 905, 5C-11"
sysContact  "Bill Roth, Amber Services, 1-512-849-3999"
```

The first part of the entry specifies the variable to be set, `sysLocation` or `sysContact`. The second part is a quoted character string representing the variable’s value. The length of this string should not exceed 256 characters. If more than one entry is in the file, the last entry is used to define the variable. If there is not an entry for a particular variable, the value is defined to be the NULL string. If there is not a quoted string after the variable name, the first word on the line is used as the value. If there is nothing after the variable name, the NULL string is assumed.

The **snmpdv1** daemon uses the defined configuration file, whether it is the default file or specified from the command line, to save and read variables. The daemon does not need to be refreshed to get these new variables.
Note: Since these variables are settable, the `snmpdv1` daemon writes to the configuration file to update these variables on a set request. If you are editing the file and a set request changes the variables, the set request could be lost when the edited file is saved. This can be avoided by shutting down the daemon to change the configuration file, or by using the `snmpinfo` command to set the variable through normal methods.

Comments are specified by a # (pound sign) character and can be located anywhere in the `snmpd.conf` file. A comment begins at the # character and continues to the end of the line.

Note: It does not matter in which order the specific configuration entries for community, traps, views, logging, `snmpd`, and smux are placed in the `snmpd.conf` file. There is no order dependency for the various entries.

Examples

1. Example of community entries in the `snmpd.conf` file:

```
# Community specifications
community public
  community private  192.100.154.7 255.255.255.255 readWrite 1.17.2
  community monitor  192.100.154.1 255.255.255.0 readWrite 1.17.2
community private oilsers
  community simple giants
community test    0.0.0.0  0.0.0.0 none
community nobody  0.0.0.0  255.255.255.255 readWrite 1.17.35
```

The first entry exemplifies the minimum required specification for a community entry. The IP address defaults to 0.0.0.0. The network mask defaults to 0.0.0.0. The permissions default to readOnly. The view defaults to the entire MIB tree. This configuration enables the `snmpdv1` agent to accept all readOnly requests under the community name `public` regardless of the IP address. Write or set requests are rejected.

The second entry limits the `snmpdv1` agent to accept readWrite requests under the community name `private` only from IP address 192.100.154.7 for MIB variables that are associated with the view name 1.17.2.

The third entry enables the `snmpdv1` agent to accept readWrite requests under the community name `monitor` from all IP addresses that start with 192.100.154, as indicated by the network mask, for all MIB variables that are associated with the view name 1.17.2.

The fourth entry sets the network mask to the default 255.255.255.255 and the permissions to the default, readOnly. This configuration enables the `snmpdv1` agent to accept readOnly requests under the community name `private` from the host named `oilers` for the entire MIB tree. The reuse of the community name `private` is independent of the usage in the second example entry.

The fifth entry sets the network mask to the default 255.255.255.255 and the default permissions to readOnly. This configuration enables the `snmpdv1` agent to accept readOnly requests for the entire MIB tree under the community name `simple` only from the host `giants`. Write or set requests are rejected.

The sixth entry causes the `snmpdv1` agent to reject all requests under the community name `test`, regardless of the IP address, because of the permission restriction of none.

The seventh entry causes the `snmpdv1` agent to reject all requests under the community name `nobody` because the network mask limits the IP address to entry 0.0.0.0, which is reserved and not available for a host.

2. Example of trap entries in the `snmpd.conf` file:

```
# Trap host notification specifications
trap traps  192.100.154.7
trap traps  129.35.39.233
trap events giants
trap public oilsers 1.2.3 be
trap private 129.35.42.2101.2.4  7e
```

AIX 5L Version 5.3 Files Reference
The first entry specifies that the `snmpdv1` agent is to notify the host with IP address 192.100.154.7 of all traps generated. The community name embedded in the trap packet will be `traps`.

The second entry specifies that the `snmpdv1` agent is to notify the host with IP address 129.35.39.233 of all traps generated. The community name embedded in the trap packet will be `traps`.

The third entry specifies that the `snmpdv1` agent is to notify the host `giants` of all traps generated. The community name embedded in the trap packet will be `events`.

The fourth entry specifies that the `snmpdv1` agent is to notify the host `oilers` of all traps generated except for the `warmStart` trap. The community name embedded in the trap packet will be `public`. The `ViewName`, 1.2.3, is ignored.

The fifth entry specifies that the `snmpdv1` agent is to notify the host 129.35.42.210 of all traps generated except the `coldStart` trap. The community name embedded in the trap packet will be `private`. The `ViewName`, 1.2.4, is ignored.

3. Examples of view entries in the `snmpd.conf` file:

```conf
# View specifications
view 1.17.2 system enterprises view
view 1.17.35
view 2.10.1 iso.3
```

The first entry associates the view name 1.17.2 with the system, enterprises, and view MIB groups. A community name that is associated with view 1.17.2 will only be associated with the MIB variables in these three groups. Thus, a host that has read permissions with this community name association can only get values for MIB variables in these specified groups.

The second and third entries configure the `snmpdv1` agent to allow access to the entire MIB tree for hosts that have access privileges associated with these specified view names.

4. Examples of logging entries in the `snmpd.conf` file:

```conf
# Logging specifications
logging /tmp/snmpdlog enabled
logging level=2 size=100000
```

These logging entries configure the `snmpdv1` agent to log messages at debug level 2 and below to the file named `/tmp/snmpdlog`. The size parameter limits the file size of the `/tmp/snmpd` log file to 100,000 bytes. When the log file reaches 100,000 bytes, the log file is rotated such that the full file is renamed to `/tmp/snmpd.0` and the new log file is named `/tmp/snmpdlog`.

5. Example of `snmpd` entries in the `snmpd.conf` file:

```conf
# snmpd parameter specifications
snmpd maxpacket=2048
snmpd querytimeout=120
```

The first `snmpd` entry limits the size of packets transmitted by the `snmpdv1` agent to 2048 bytes. The second entry sets the `querytimeout` parameter to 120 seconds. This configures the `snmpdv1` agent to query all the interfaces known to the TCP/IP kernel every two minutes for status changes.

6. Examples of smux entries in the `snmpd.conf` file:

```conf
# smux configuration
smux 1.3.6.1.4.1.2.3.1.2.2 #gated
```

This smux entry configures the `snmpdv1` agent to allow the SMUX association only the `gated` SMUX peer client with no authentication. The SMUX peer must be running on the local host.

```conf
# smux configuration
smux 1.3.6.1.4.1.2.3.1.2.2 private #gated
```

This smux entry configures the `snmpdv1` agent to allow the SMUX association only the `gated` SMUX peer client having the password `private`. The SMUX peer must be running on the local host.

```conf
# smux configuration
smux 1.3.6.1.4.1.2.3.1.2.2 private 0.0.0.0 0.0.0.0
```

This smux entry configures the `snmpdv1` agent to allow the SMUX association only the `gated` SMUX peer client having the password `private`. The SMUX peer can be running on any host.
This smux entry configures the `snmpdv1` agent to allow the SMUX association only the **gated** SMUX peer client having the password `private`. The gated SMUX peer must be running on the host with IP address 192.100.154.7

This entry configures the `snmpdv1` agent to allow the SMUX association only the **gated** SMUX peer client having the password `private`. The gated SMUX peer can be running on any host in the network defined by 192.100.154.

**Note:** The SMUX peer client object identifier must be unique. Only one form of the preceding examples of smux entries for the gated SMUX peer client can be in the `snmpd.conf` file.

7. Example of sysLocation and sysContact entries in the `snmpd.conf` file:

```
# Definitions for sysLocation and sysContact
sysLocation   "Austin, Texas, USA, XYZ, Bld 905, 5C-11"
sysContact    "Bill Roth, Amber Services, 1-512-849-3999"
```

These entries set the value for the `sysLocation` and `sysContact` variables.

---

**Related Information**

The [snmpdv1](https://www.example.com) command.

The [gated](https://www.example.com) daemon.

Problem Determination for the SNMP Daemon, [SNMP trap processing](https://www.example.com) [SNMP daemon troubleshooting](https://www.example.com) in *Networks and communication management*.

[SNMP daemon configuration](https://www.example.com) in *Networks and communication management*.

---

**snmpd.boots File**

### Purpose

Provides the boot and engine ID information for the `snmpdv3` agent.

### Description

The `snmpt.boots` file provides the boot and engine ID information for the `snmpdv3` agent. The file contains two elements: an `engineID`, and, `engineBoots`, the number of times that the `snmpv3` daemon has been started.

### Syntax

```
engineID engineBoots
```

Where:

- **engineID**
  
  A string of 2 to 64 (must be an even number) hexadecimal digits. The engine identifier uniquely identifies the agent within an administrative domain. By default, the engine identifier is created using a vendor-specific formula and incorporates the IP address of the agent. However, a customer can choose to use any engine identifier that is consistent with the `snmpEngineID` definition in RFC 2271 and that is also unique within the administrative domain.
The first 8 hex digits represent a vendor enterprise ID obtained from the Internet Assigned Numbers Authority (IANA). For IBM, this is 00000002. The last 16 hex digits are determined by vendor formula. For IBM, this is:

- The first two hex digits indicate the content of the next fourteen hex digits.
- 00 indicates the next six hex digits are zeros, followed by the IP address of the agent in the last eight hex digits.
- 01 indicates the next six hex digits contain a timestamp, followed by the IP address of the agent in the last eight hex digits.

For the agent, we always use the one without a timestamp, so the engineID for an SNMP agent at IP address 9.67.113.10 would be:

00000002 00000000 09 43 71 0A

(Spaces added to improve readability)

**engineBoots**

The number of times (in decimal) the agent has been restarted since the engineID was last changed.

**Notes:**
1. engineID and engineBoots must be specified in order and on the same line.
2. Comments are specified in the file by starting the line with either an asterisk (*) or a pound sign (#).
3. No comments are allowed between engineID and engineBoots values.
4. Only the first non-comment line is read. Subsequent lines are ignored.

**Example**

The first string of numbers is the **engineID**, the second string is the number of times the **snmpv3** daemon has been started.

00000002000000000903E65F 0000000003

**Files**

*etc/snmpd.boots* Provides boot and **engineID** information.

**Related Information**

The **snmpdv3** command.

Problem Determination for the SNMP Daemon, [SNMP trap processing](#), [SNMP daemon troubleshooting](#) in *Networks and communication management*.

**snmpdv3.conf File**

**Purpose**

Defines a sample configuration file for the **snmpdv3** agent.

**Description**

An entry must be contained on one line (i.e., the newline character will be treated as the end of an entry). All of the entry definitions require that all fields on the entry are specified, either with a specific value or a dash (-) to denote the default value. If an error is detected processing an entry and no appropriate default
value can be assumed, the entry will be discarded. Statements in the file are not order-dependent. However, if more than one DEFAULT_SECURITY statement is found, the last one in the file is the one that is used.

**General Usage Rules**
- All values for an entry must be on the same line.
- All keys need to be regenerated using the pwtokey command in order for these sample entries to actually be used.
- In this sample: Keys are generated for use with engineID 00000002000000000943714F.
- Authentication keys were generated with password of username + password, such as u1password.
- Privacy keys were generated with password of username + privpass, such as u1privpass.
- Entries defined to use encryption support, which is available only as a separately orderable feature on the base AIX product, are included below but commented out.

Comments may be entered in the **snmpdv3.conf** file, with the following restrictions:
- Comments must begin with the pound sign (#) or asterisk (*).
- Comments must begin in column 1. This allows the pound sign and asterisk to be used in names of users, views, etc.

**USM_USER entries**
Defines a user for the User-based Security Model (USM). Format is:

```
userName engineID authProto authKey privProto privKey keyType storageType
```

where

**userName**
Indicates the name of the user for the User-based Security Model (USM) and must be unique to the SNMP agent. The `userName` is used as the security name for the User-based Security Model. The contents of this field will be used as the `securityName` value for other entries (such as the VACM_GROUP entry) when the `securityModel` is USM.
Valid value is:
- An octet string of 1 to 32 octets (characters).

There is no default value.

**engineID**
Indicates the `engineID` of the authoritative side of the message. The `engineID` for the AIX SNMP agent is determined at agent initialization. It is either read in from the **SNMPD.BOOTS** file or it is generated automatically and stored in the **SNMPD.BOOTS** file. It can be retrieved dynamically by issuing a `get` request for object `snmpEngineID`. For `get`, `getbulk`, `set`, `response`, and `trap` messages, the authoritative side is the SNMP agent. For inform messages, the authoritative side is the notification receiver.

**Note:** AIX will not support informs. `engineID` is defined in RFC 2271.
Valid values are:
- An octet string of 1 to 32 octets (2 to 64 hex digits).
- A ` '-' ` (dash) indicates the default value.

The default value is the local SNMP agent's `engineID`.

**authProto**
Indicates the authentication protocol to be used on authenticated messages on behalf of this user. Valid values are:
- **HMAC-MD5** - indicates HMAC-MD5.
- **HMAC-SHA** - indicates HMAC-SHA.
- **none** - indicates no authentication is to be done.
The default value is HMAC-MD5 (if an authentication key is specified; if no authentication key is specified, no authentication can be done for messages to/from this user).

**authKey**
Indicates the authentication key to be used in authenticating messages on behalf of this user. This field will be ignored when **authProto** is specified as none. The **keyType** field will indicate whether the key is localized or non-localized. Valid values are:
- An octet string of 16 bytes (32 hex digits) when **authProto** is HMAC-MD5.
- An octet string of 20 bytes (40 hex digits) when **authProto** is HMAC-SHA.
- A '-' (dash) indicates the default.

The default value is no key, indicating no authentication.

**privProto**
Indicates the privacy protocol to be used on encrypted messages on behalf of this user. Privacy can be requested only if authentication is also requested. If authentication is not requested, this field is ignored. Valid values are:
- **DES** - indicates CBC-DES (only with the additional encryption product).
- **none** - indicates no privacy.
- A '-' (dash) indicates default.

The default value is no privacy. No encryption will be done on messages to/from this user.

**privKey**
The privacy key to be used in authenticating messages to and from this user. This field will be ignored when **privProto** is specified or defaulted as **none**. The **keyType** field will indicate whether the key is localized or non-localized. Privacy can be requested only if authentication is also requested. If authentication is not requested, this field is ignored. The privacy key and the authentication key are assumed to have been generated using the same authentication protocol (HMAC-MD5 or HMAC-SHA). Valid values are:
- An octet string of 16 bytes (32 hex digits) if the key is localized or if the key is non-localized and the **authProto** is HMAC-MD5.
- An octet string of 20 bytes (40 hex digits) if the key is non-localized and the **authProto** is HMAC-SHA.
- A '-' (dash) indicates default.

Default value is no key, indicating no encryption.

**keyType**
Indicates whether the keys defined by **authKey** and **privKey** are localized or non-localized. Localized indicates that they have been generated with the appropriate engineID making the key usable only at one snmpEngine. Non-localized indicates the key may be used at different snmpEngines. The **authKey** and **privKey**, if both are specified, must both be localized or both be non-localized. This field is ignored if no authentication or privacy is requested. Valid values are:
- **L** - indicates keys are localized.
- **N** - indicates keys are non-localized.
- A '-' (dash) indicates default Default value is localized.

**storageType**
Indicates the type of storage in which this definition is to be maintained. **StorageTypes** are defined in RFC1903. Valid values are:
- **nonVolatile** - indicates the entry definition will persist across reboots of the SNMP agent, but it can, however, be changed or even deleted by dynamic configuration requests.

- **permanent** - indicates the entry definition will persist across reboots of the SNMP agent; it can be changed but not deleted by dynamic configuration requests.

- **readonly** - indicates the entry definition will persist across reboots of the SNMP agent; it cannot be changed or deleted by dynamic configuration requests. **readOnly** is not permitted if the authentication protocol is not 'none' (because keys must be changeable per RFC 2274 definition of **usmUserStorageType**).

- `'-` (dash) - indicates default.

Default value is non-volatile.

**VACM_GROUP entries**

Defines a security group (made up of users or communities) for the View-based Access Control Model (VACM). Format is:

\[
\text{groupName} \text{ securityModel} \text{ securityName} \text{ storageType}
\]

where:

- **groupName**
  Indicates the group name for the View-based Access Control Model (VACM) and must be unique to the SNMP agent. Valid value is:
  - An octet string of 1 to 32 octets (characters).
  
  There is no default value.

- **securityModel**
  Indicates the SNMP security model for this entry. When an SNMP message comes in, the **securityModel** together with the **securityName** are used to determine to which group the user (or community) represented by the **securityName** belongs. Valid values are:
  - 'SNMPv1' - indicates community-based security using SNMPv1 message processing.
  - 'SNMPv2c' - indicates community-based security using SNMPv2c message processing.
  - 'USM' - indicates User-based Security Model. A `'-` (dash) - indicates default. Default value is 'USM'.

- **securityName**
  Indicates a member of this group. For community-based security, it will be a community name. For the User-based Security Model, it will be a user name. Valid values are:
  - An octet string of 1 to 32 octets (characters) indicating a USM userName when **securityModel** is USM.
  - An octet string of 1 to 32 octets (characters) indicating a community Name when **securityModel** is 'SNMPv1' or 'SNMPv2c'.

  There is no default value.

- **StorageType**
  As defined above on the **USM_USER** definition.

**VACM_VIEW entries**

Defines a particular set of MIB data, called a view, for the View-based Access Control Model. Format is:

\[
\text{viewName} \text{ viewSubtree} \text{ viewMask} \text{ viewType} \text{ storageType}
\]

where:

- **viewName**
  Indicates the textual name of the view for the View-based Access Control Model. View
names do not need to be unique. Multiple entries with the same name together define one view. However, the **viewname**, together with the **subtree** object ID, must be unique to an SNMP engine. Valid values are:

- An octet string of 1 to 32 octets (characters).

There is no default value.

**viewSubtree**

Indicates the MIB object prefix of the MIB objects in the view. Valid values are:

- An object id of up to 128 sub-OIDs.
- A textual object name (or object prefix).
- A combination of textual object name followed by numeric sub-OIDs. The name must be found within the compiled MIB or in the logical extension to the MIB, the **MIBS.DATA** file.

There is no default value.

**viewMask**

Indicates a mask that specifies which of the sub-OIDs in the subtree are relevant. See RFC2275 for a definition of the viewMask. Valid values are:

- A hex string of up to 16 octets (up to 128 bits) where each bit indicates whether or not the corresponding sub-OID in the subtree is relevant.
- A '-' (dash) - indicates default.

The default value is a mask of all (meaning all sub-OIDs are relevant).

**viewType**

Indicates the type of the view definition. Valid values are:

- **included** - indicating the MIB objects identified by this view definition are within the view.
- **excluded** - indicating the MIB objects identified by this view definition are excluded from the view.
- A '-' (dash) - indicates default.

The default value is **included**.

**storageType**

As defined above on the **USM_USER** definition.

**VACM_ACCESS** entries

Identifies the access permitted to different security groups for the View-based Access Control Model. Format is:

```
groupName contextPrefix contextMatch securityLevel, securityModel readView writeView notifyView storageType
```

where:

**groupName**

Indicates the group name for the View-based Access Control Model (VACM) for which access is being defined. Valid values are:

- An octet string of 1 to 32 octets (characters).

There is no default value.

**contextPrefix**

Indicates an octet string to be compared with the incoming contextName if the value specified for the contextMatch field is **prefix**. Note, however, that the SNMP agent in AIX supports MIB objects in only the local (null) context. Valid values are:

- An octet string of 1 to 32 octets (characters).
- A '-' (dash) - indicates default.

The default value is the null context (""").

**contextMatch**
Indicates whether the incoming contextName must be compared with (and match exactly) the entire contextName or whether only the first part of the contextName (up to the length of the indicated value of the contextPrefix) must match. Valid values are:
- **exact** - indicates entire contextName must match.
- **prefix** - indicates only the prefix of the contextName must match.
- A '-' (dash) - indicates the default.

The default value is exact.

**securityLevel**
Indicates the securityLevel for this entry. Used in determining which access table entry to use. Valid values are:
- **noAuthNoPriv** or 'none' - indicates no authentication or privacy protocols applied.
- **AuthNoPriv** or 'auth' - indicates authentication protocols applied but no privacy protocol is applied.
- **AuthPriv** or 'priv' - indicates both authentication and privacy protocols applied (If the additional encryption pack is not applied, this level can be configured but cannot actually be used).
- A '-' (dash) - indicates default.

The default value is **noAuthNoPriv**.

**securityModel**
Indicates the SNMP security model for this entry. Used in determining which access table entry to use. Valid values are:
- **SNMPv1** - indicates community-based security using SNMPv1 message processing.
- **SNMPv2c** - indicates community-based security using SNMPv2c message processing.
- **USM** - indicates User-based Security Model.
- A '-' (dash) - indicates default.

The default value is **USM**.

**readView**
Indicates the name of the view to be applied when read operations (get, getnext, getbulk) are performed under control of this entry in the access table. Valid values are:
- An octet string of 1 to 32 octets (characters) identifying a view defined by a VACM_VIEW definition.
- A '-' (dash) - indicates default.

The default value is no view; no **readView** defined for members of this group.

**writeView**
Indicates the name of the view to be applied when write operations (set) are performed under control of this entry in the access table. Valid values are:
- An octet string of 1 to 32 octets (characters) identifying a view defined by a VACM_VIEW definition.
- A '-' (dash) - indicates default.

The default value is no view; no **writeView** defined for members of this group.

**notifyView**
Indicates the name of the view to be applied when notify operations (traps or informs) are performed under control of this entry in the access table. Valid values are:
An octet string of 1 to 32 octets (characters) identifying a view defined by a VACM_VIEW definition.

A '-' (dash) - indicates default.

Default value is no view; no notifyView defined for members of this group.

**NOTIFY entries**

Identifies management targets to receive notifications. Format is:

```
notifyName tag type storageType
```

where:

- **notifyName**
  - Is a locally unique identifier for this notify definition. Valid values are:
    - An octet string of 1 to 32 octets (characters)
  - There is no default value.

- **tag**
  - Indicates a tag value to be compared with the values in the tagLists defined in the snmpTargetAddrTable (either on TARGET_ADDRESS entries or via dynamic configuration). For each match of this tag with a value in the tagLists defined in the snmpTargetAddrTable, a notification may be sent. See RFC2273 for a definition of SnmpTagValue. Valid values are:
    - An octet string of 1 to 255 octets (characters). No delimiters are allowed.
    - A '-' indicates the default.
  - Default value is no tag value.

- **type**
  - Indicates which type of notification should be generated. Valid values are:
    - trap - an unconfirmed notification; notification sent with trap PDUs.
    - A '-' (dash) - indicates the default.
  - Default value is trap. inform type traps are not supported on AIX.

**TARGET_ADDRESS**

Defines a management application's address and parameters to be used in sending notifications. Format is:

```
targetAddrName tDomain tAddress tagList targetParams timeout retryCount storageType
```

where:

- **targetAddrName**
  - Indicates a locally unique identifier for this target address definition. Valid values are:
    - An octet string of 1 to 32 octets (characters).
  - There is no default value.

- **tDomain**
  - Indicates the transport type of the address indicated by tAddress. Valid values are:
    - UDP - for UDP datagrams.
    - A '-' (dash) - for the default value.
  - Default value is UDP.

- **tAddress**
  - Indicates the transport address to which notifications are sent. Valid values are:
    - A 1- to 21- octet string indicating the IP address and optionally the UDP port.
  - Form is
**ip_address:port**

IP address must be specified as a.b.c.d where a, b, c and d are in the range of 0 to 255. The port, if specified, must be in the range of 1 to 65535. Example:

9.37.84.48:162

The IP address may not be defaulted, but the port, if not specified, will default to 162.

**tagList**
Indicates a list of tag values which are used to select target addresses for a notification operation. The AIX implementation will support, via the configuration file, only one tag in a tagList. Because informs are not supported, there is no particular value in supporting multi-tag tagLists. RFC2273 contains the complete definition of SnmpTagList and SnmpTagValue. The AIX implementation accepts as valid values:
- An octet string of 1 to 255 octets (characters). No delimiters are allowed.
- '-' indicates the default.

The default value is an empty list.

**targetParams**
Indicates a TARGET_PARAMETERS paramsName value that indicates which security and message processing is to be used in sending notifications to this target. Valid values are:
- An octet string of 1 to 32 octets (characters)

There is no default value.

**timeout**
Indicates the expected maximum round trip time for communicating with this target address (in 1/100ths of a second). timeout is used only for inform type notifications; it is not used for traps. Since only traps are supported on AIX, only the default value is accepted. Valid values are:
- An integer in the range of (0..2147483647) specifying the number of hundredths of a second for the timeout. Note, however, that this value is not used for notifications of type trap.
- '-' (dash) indicating the default.

Default value is 0, meaning no timeout value.

**retryCount**
Indicates the number of retries to be attempted when a response is not received for a generated message. retryCount is used only for inform type notifications; it is not used for traps. Since only traps are supported on AIX, only the default value is accepted. Valid values are:
- An integer in the range of (0 to 255), indicating the number of retries to be attempted. Note, however, that this value is not used for notifications of type trap.
- A '-' (dash) indicating the default.

Default value is 0, meaning no retry.

**TARGET_PARAMETERS**
Defines the message processing and security parameters to be used in sending notifications to a particular management target. Format is:

```
paramsName mpModel securityModel securityName securityLevel storageType
```

where:

**paramsName**
A locally unique identifier for this target parameters definition. Valid values are:
mpModel
The message processing model to be used in sending notifications to targets with this parameter definition. Valid values are:
- SNMPv1 - indicates SNMPv1.
- SNMPv2c - indicates SNMPv2c.
- SNMPv3 - indicates SNMPv3.
There is no default value.

securityModel
Indicates the security model to be used in sending notifications to targets with this parameter definition. Valid values are:
- SNMPv1 indicates SNMPv1.
- SNMPv2c Indicates SNMPv2c.
- USM indicates User-based Security Model.
There is no default value.

securityName
Identifies the principal (user or community) on whose behalf SNMP messages will be generated using this parameter definition. For community based security, this would be a community name. For USM, this would be a user name. Valid values are:
- An octet string of 1 to 32 octets (characters).
There is no default value.

securityLevel
Indicates the security level to be used in sending notifications to targets with this parameter definition. Valid values are:
- noAuthNoPriv or none - indicates no authentication or privacy. protocols applied.
- AuthNoPriv or auth - indicates authentication protocols applied but no privacy protocol is applied.
- AuthPriv or priv - indicates both authentication and privacy protocols applied. (If the additional encryption pack is not applied, this level can be configured, but not actually used.)
- '-' (dash) - indicates default.
Default value is noAuthNoPriv.

COMMUNITY
Defines a community for community-based security. Format is:
communityName securityName securityLevel netAddr netMask storageType

where:

communityName
Indicates a community name for community-based security (SNMPv1 or # SNMPv2c). Valid values are:
- An octet string of 1 to 32 octets (characters).
There is no default value.

securityName
Indicates a securityName defined for this communityName. The securityName is the more generic term for the principal (user or community) for which other entries, such as
VACM_GROUP and TARGET_PARAMETERS, are defined. Typically, the securityName would match communityName or, at least, there would be a one-to-one correspondence between securityName and communityName. (Until the community MIB support is implemented, the community name must match the securityName exactly.) Valid values are:

- An octet string of 1 to 32 octets (characters).
- `-` (dash) - indicates default.

The default value is securityName equal to the specified communityName.

**securityLevel**

Indicates the security level to be applied when processing incoming or outgoing messages with this community name.

**Note:** When the communityMIB is implemented, authNoPriv will also be a valid level of security, but at the moment, it will be rejected because there is no way to store a securityLevel to be associated with a communityName. When that happens, the following will be added to the list of valid values below:

- authNoPriv or auth - indicates authentication protocols applied.

Note that no additional authentication checking is done by specifying auth. Authentication still involves verifying that the community name is being used by an IP address for which it has been defined and using the views defined for that entry. However, allowing the specification of auth here does allow the system administrator to define a different set of views to be used if the same community name is defined with two different securityNames (each with a different securityLevel)

Valid values are:

- noAuthNoPriv or none - indicates no authentication or privacy protocols applied.
- `-` (dash) - indicates default.

Default value is noAuthNoPriv. Encryption is not supported on SNMPv1/SNMPv2c messages.

**netAddr**

A network IP address in dotted decimal notation indicating the range of addresses for which this community name may be used. Valid values are:

- A network address in the form of a.b.c.d, where a, b, c and d are in the range of 0 to 255. (Note, not all four octets are required. Also, 255.255.255.255 is not a valid network address.)

There is no default value.

**netMask**

An IP address mask to be logically ANDed with the origin address of the incoming SNMP message. If the resulting value equals the value specified for netAddr, the incoming message is accepted. Valid values are:

- A network address in the form of a.b.c.d, where a, b, c and d are in the range of 0 to 255. (Not all four octets are required.)

There is no default value.

**storageType**

As defined above on the USM_USER definition (Note, until the community MIB is implemented, storage type values other than readOnly will be treated as readOnly; i.e., they cannot be changed dynamically.)

**DEFAULT_SECURITY**

Identifies the default security posture to be configured for the SNMP agent; additional security
definitions defined by the use of the preceding eight entry definition types augment any default security configurations defined as a result of the **DEFAULT_SECURITY** statement. Format is:

```
securityPosture password privacy
```

where:

**securityPosture**
- Indicates the default security posture to be configured for the SNMP agent, as defined by Appendix A of RFC 2275 (and outlined below). Valid values are:
  - **minimum-secure** - indicates the SNMP agent will be configured with the least secure default configurations
  - **semi-secure** - indicates the SNMP agent will be configured with moderately secure default configurations.
  - **no-access** - indicates the SNMP agent will be configured with no default configurations.

The default value is **no-access**.

**password**
- Indicates the password to be used to generate authentication and privacy keys for user 'initial' In the case that **no-access** is specified as the **securityPosture**, this keyword is ignored. Valid values are:
  - An octet string of 8 to 255 octets (characters).
  - '-' (dash) - indicating the default.

Default value is no password. Default only accepted if **securityPosture** is **no-access**.

**privacy**
- Indicates whether or not encryption is to be supported for messages on behalf of user 'initial'. Valid values are:
  - **Yes** - indicates privacy is supported for user 'initial' (only with the additional encryption product).
  - **No** - indicates privacy is not supported for user 'initial'.
  - '-' (dash) - indicates default value.

Default value is no. If **no-access** is selected as the security posture, this value will be ignored.

Default security definitions based on the selected security posture:

**no-access**
- No initial configurations are done.

**semi-secure**
- The default (null) context is configured. If privacy is not requested, a default user is configured as if the following USM_USER entry had been specified. USM_USER initial- HMAC-MD5 none - N permanent where ### indicates the key generated from the password specified on the DEFAULT_SECURITY entry. If privacy is requested (and available with the additional encryption product), a default user is configured as if the following USM_USER entry had been specified: USM_USER initial - HMAC-MD5 ### DES ### N permanent where ### indicates the key generated from the password specified on the DEFAULT_SECURITY entry.

A default group is configured as if the following VACM_GROUP entry had been specified:

```
VACM_GROUP initial USM initial readOnly.
```

Three default access entries are configured as if the following VACM_ACCESS entries had been specified:

```
VACM_ACCESS initial - exact none. USM restricted - restricted readOnly.
VACM_ACCESS initial - exact auth. USM internet internet internet internet readOnly.
```
Two default MIB views are configured as if the following VACM_VIEW entries had been specified:

VACM_VIEW internet internet - included readOnly
VACM_VIEW restricted system - included readOnly
VACM_VIEW restricted snmp - included readOnly
VACM_VIEW restricted snmpEngine - included readOnly
VACM_VIEW restricted snmpMPDStats - included readOnly
VACM_VIEW restricted usmStats - included readOnly

minimum-secure
The default (null) context is configured. If privacy is not requested, a default user is configured as if the following USM_USER entry had been specified: USM_USER initial - HMAC-MD5 ### none - N permanent where ### indicates the key generated from the password specified on the DEFAULT_SECURITY entry.

If privacy is requested (and available with the additional encryption product), a default user is configured as if the following USM_USER entry had been specified: USM_USER initial - HMAC-MD5 ### DES ### N permanent where ### indicates the key generated from the password specified on the DEFAULT_SECURITY entry.

A default group is configured as if the following VACM_GROUP entry had been specified: VACM_GROUP initial USM initial readOnly.

Three default access entries are configured as if the following VACM_ACCESS entries had been specified:

VACM_ACCESS initial - exact priv USM restricted - restricted readOnly
VACM_ACCESS initial - exact auth USM internet internet internet internet readOnly
VACM_ACCESS initial - exact priv USM internet internet internet internet readOnly

Two default MIB views are configured as if the following VACM_VIEW entries had been specified:

VACM_VIEW internet internet - included readOnly
VACM_VIEW restricted internet - included readOnly

logging
Directs logging from the configuration file. Format is:

logging file=/path/filename enabled|disabled
logging size=<limit> level=<debug_level>

There can be no white spaces around the "=" in the file, size and level fields where <path/filename> specifies the complete path and filename of the log file. Valid values are: An octet string of 1 to 255 octets (characters). Default value is /var/tmp/snmpdv3.log enabled|disabled. Valid values are: 'enabled' - turns logging on; 'disabled' - turns logging off. Default value is 'enabled'.

<limit>
Specifies the maximum size in bytes of the specified logfile. Valid values are: '0' - meaning unlimited. An unsigned integer number in the unit of byte. Default value is 0.

<debug level>
Specifies the logging level. Valid values are: # 0, 1, 2, 3, or 4 Default value is 0.

smux entry
Sets the smux peer configuration parameters # Format is:

smux <client OIdentifier> <password> <address> <netmask>

Fields to the right of <client OIdentifier> are optional, with the limitation that no fields to the left of a specified field are omitted. Where <client OIdentifier> defines the unique object identifer in
dotted decimal notation of the SMUX peer client. Valid values are: An unique object identifier in
dotted decimal notation up to 128 sub-OIDs of that SMUX peer. There is no default value.

 `<password>`
Specifies the password that snmpd requires from the SMUX peer client to authenticate the SMUX
association. If no password is specified, there is no authentication for the SMUX association. Valid
values are: An octet string of 8 to 255 octets (characters). Default value is null string

 `<address>`
Identifies the host on which the smux peer client is executing. Valid values are: A host name of 1
to 80 characters or IPv4 address. IP address must be specified as a.b.c.d where a, b, c and d are
in the range of 0 to 255. Default value is 127.0.0.1

 `<netmask>`
Specifies the network mask. Valid values are: network mask must be specified as a.b.c.d where a,
b, c and d are in the range of 0 to 255. Default value is 255.255.255.255.

 smux 1.3.6.1.4.1.2.3.1.2.1.2  gated_password # gated

Any SNMP agent configuration entries added by dynamic configuration (SET) requests get added to the
end of the snmpdv3.conf file.

Related Information
The snmpdv3, clsnmp, pwtokey, and pwchange commands.
The /etc/clsnmp.conf file.

Problem Determination for the SNMP Daemon, SNMP trap processing, SNMP daemon troubleshooting in
Networks and communication management.

SNMP daemon configuration in Networks and communication management.

snmpmibd.conf File

Purpose
Defines the configuration parameters for snmpmibd dpi2 sub-agent.

Description
The snmpmibd.conf file provides the configuration information for the snmpmibd dpi2 sub-agent. This
file can be changed while the snmpmibd dpi2 sub-agent is running. If the refresh command is issued,
the snmpmibd dpi2 sub-agent will reread this configuration file. The snmpmibd dpi2 sub-agent must be
under System Resource Control (SRC) for the refresh command to force the reread. To perform a reread,
as root user, run:

 # refresh -s snmpmibd

Keywords
The directives are specified in the form of `<keyword>=<value>`. The keyword is case-insensitive. The value
passed is also case-sensitive.

logFilename
The name of the most recent log file. Less recent log files have the number 1 to (n - 1) appended
to their names. The larger the number, the less recent the file.

logFileSize
The size of log files in K bytes. Maximum size of a log file. When the size of the most recent log
file reaches this value, it is renamed and a new log file is created.
numLogFiles
The number of log files desired. The maximum value for numLogFiles is 4. A new file is created when the size of the log file is equal or more than the size specified by the keyword logFileSize. When the number of log files reaches the numLogFiles the log files start rotating.

tracelevel
The tracing/debug level.

8 = DPI level 1
16 = DPI level 2
32 = Internal level 1
64 = Internal level 2
128 = Internal level 3

Add the numbers for multiple trace levels

Example
logFileName=/usr/tmp/snmpmibd.log
logFileSize=0
numLogFiles=0
tracelevel=0

File
/etc/snmpmibd.conf Defines the configuration parameters for snmpmibd dpi2 sub-agent.

Related Information
The snmpmibd and refresh commands.
The snmpdv3.conf file documentation.

socks5c.conf File

Purpose
Contains mappings between network destinations and SOCKSv5 servers.

Description
The /etc/socks5c.conf file contains basic mappings (between network destinations, hosts or networks, and SOCKSv5 servers) to use when accessing network destinations. It is an ASCII file that contains records for server mappings. Text that follows a pound character (#) is ignored until the end of the line. Each record is on a single line in the following format:

destination [/prefixlength] server [:port]

You must separate the fields with whitespace. Records are separated by new line characters. The fields and modifiers in a record have the following values:

destination Specifies a network destination. The destination variable may be either a name fragment or a numeric address (with optional prefixlength). If destination is an address, it may be either IPv4 or IPv6.

prefixlength If specified, indicates the number of leftmost (network order) bits of an address to use when comparing to this record. It is valid only if destination is an address. If not specified, all bits are used in comparisons.
server Specifies the SOCKSv5 server associated with destination. If server is NONE (must be all uppercase), this record indicates that target addresses matching destination should not use any SOCKSv5 server; instead, it should be contacted directly.

port If specified, indicates the port to use when contacting server.

If a name fragment destination is present in /etc/socks5c.conf, all target addresses in SOCKSv5 operations will be converted into hostnames for name comparison (in addition to numeric comparisons with numeric records). The resulting hostname is considered to match if the last characters in the hostname match the specified name fragment.

When using this configuration information to determine the address of the appropriate SOCKSv5 server for a target destination, the best match is used. The best match is defined as follows:

- If destination is numeric, the most bits in the comparison (i.e., largest prefixlength) are used.
- If destination is a name fragment, the most characters in the name fragment are.

When both name fragment and numeric addresses are present, all name fragment entries are better than numeric address entries.

The following two implicit records are assumed as defaults for all destinations not specified in /etc/socks5c.conf:

```
0.0.0.0/0 NONE #All IPv4 destinations; no associated server.
::/0 NONE #All IPv6 destinations; no associated server.
```

SOCKS5C_CONFIG Environment Variable
The SOCKS5C_CONFIG environment variable enables the SOCKS library. To enable the library and to indicate that it uses the socks5c.conf file, you must set and export the variable to the pathname of the file, which is /etc/socks5c.conf. However, you can use a different configuration file by setting SOCKSSC_CONFIG to the pathname of that file. If the specified file is not found, then by default the socks5c.conf file is used as a configuration file. If you set this variable to NULL, then SOCKS is not used and traditional network operations occur instead.

Security
Access Control: This file should grant read (r) access to all users and grant write (w) access only to the root user.

Examples
```
#Sample socks5c.conf file
9.0.0.0/8 NONE #Direct communication with all hosts in the 9 network.
129.35.0.0/16 soxl.austin.ibm.com
ibm.com NONE #Direct communication will all hosts matching "ibm.com" (e.g. "aguila.austin.ibm.com")
```

Related Information
The socks5tcp_connect subroutine.

space File

Purpose
Describes the format of a disk space requirements file.
Description
The space file is an ASCII file that gives information about disk space requirements for the target environment. It defines the maximum additional space a package requires (for example, for files that are installed with the installf command).

The generic format of a line in this file is:

```
pathname  blocks  inodes
```

Definitions for the fields are as follows:

```
pathname          Specifies a directory name which may or may not be the mount point for a filesystem. Names that do not begin with a slash (/) indicate relocatable directories.
blocks            Defines the number of disk blocks required for installation of the files and directory entries contained in the pathname (using a 512-byte block size).
inodes             Defines the number of inodes required for installation of the files and directory entries contained in the pathname.
```

Examples

```
# extra space required by config data which is
# dynamically loaded onto the system
data 500 1
```

.srf File

Purpose
Contains all the text components with hypertext information embedded.

Description
The .srf file is one of several intermediate files produced for each document by InfoCrafter. The .srf file is a binary file that contains all the text components with hypertext link information embedded.

Files

```
.srf
```

Contains text components with embedded linking information.

Related Information

The .fig file.

streamcmds File

Purpose
Contains auditstream commands.

Description
The /etc/security/audit/streamcmds file is an ASCII template file that contains the stream mode commands that are invoked when the audit system is initialized. The path name of this file is defined in the stream stanza of the /etc/security/audit/config file.
This file contains command lines, each of which is composed of one or more commands with input and output that may be piped together or redirected. Although the commands usually are one or more of the audit system commands (\texttt{auditcat}, \texttt{auditpr}, and, \texttt{auditselect}), this is not a requirement. The first command, however, should be the \texttt{auditstream} command.

When the audit system is initialized, the \texttt{audit start} command runs each command. No path name substitution is performed on $\texttt{Strail}$ or $\texttt{Sbin}$ strings in the commands.

\textbf{Security}

Access Control: This file should grant read (r) access to the root user and members of the audit group, and write (w) access to the root user only.

\textbf{Examples}

1. To read all records from the audit device, select and format those that involve unsuccessful events, and print them on a line printer, include the following in the \texttt{/etc/security/audit/streamcmds} file:

   \texttt{/usr/sbin/auditstream | /usr/sbin/auditselect -e \ "result == FAIL" | /usr/sbin/auditpr -v > /dev/lpr0}

   This command is useful for creating a hard-copy trail of system security violations.

2. To read all records from the audit device that have audit events in the \texttt{authentication} class, format them, and display them on the system console. Include the following in the \texttt{/etc/security/audit/streamcmds} file:

   \texttt{/usr/sbin/auditstream -c authentication | | /usr/sbin/auditpr -t0 -v > /dev/console}

   This command allows timely auditing of user authentication events.

\textbf{Files}

\begin{itemize}
  \item \texttt{/etc/security/audit/streamcmds} \hspace{1cm} Specifies the path to the file.
  \item \texttt{/etc/security/audit/config} \hspace{1cm} Contains audit system configuration information.
  \item \texttt{/etc/security/audit/events} \hspace{1cm} Contains the audit events of the system.
  \item \texttt{/etc/security/audit/objects} \hspace{1cm} Contains audit events for audited objects (files).
  \item \texttt{/etc/security/audit/bincmds} \hspace{1cm} Contains \texttt{auditbin} backend commands.
\end{itemize}

\textbf{Related Information}

The \texttt{audit} command, \texttt{auditcat} command, \texttt{auditpr} command, \texttt{auditselect} command.

\texttt{Setting Up Auditing} in \textit{Operating system and device management}.

\texttt{Auditing overview} Security Administration in \textit{Operating system and device management}.

\textbf{sysck.cfg File}

\textbf{Purpose}

Contains file definitions for the trusted computing base.

\textbf{Description}

\textbf{Note:} The \texttt{sysck} command does not update this file. It is only updated by the \texttt{tcbck} command.
The /etc/security/sysck.cfg file is a stanza file that contains definitions of file attributes for the trusted computing base. The name of each stanza is the pathname of a file, followed by a : (colon). Attributes are in the form Attribute=Value. Each attribute is ended with a new-line character, and each stanza is ended with an additional new-line character.

Each stanza can have one or more of the following attributes, and must have the type attribute:

- **acl**  
  Defines the access control list of the file, including the SUID, SGID, and SVTX bits. The value is the Access Control List, in the format described in Access control lists in Operating system and device management.

- **class**  
  Defines a group of files for checking, deleting, or updating. A file can be in more than one class. The value is the ClassName [ClassName] parameter.

- **checksum**  
  Defines the checksum, as computed with the `sysck checksum` program. This attribute is valid only for regular files. The value is the output of the `sum -r` command, including spaces.

- **group**  
  Defines the group name or numeric group ID, expressed as the GroupName or GroupID parameter.

- **links**  
  Defines the absolute paths that have hard links to this object. The value must be an absolute pathname, expressed as the Path [Path ...] parameter.

- **mode**  
  Defines the file mode, expressed as the Flag, Flag ..., PBits parameters. The Flag parameter can contain the SUID, SGID, SVTX, and tcb mode attributes. The Pbits parameter contains the base file permissions, expressed either in octal form, such as 640, or symbolic form, such as rw-,r—, r—. The order of the attributes in the Flag parameter is not important, but base permissions must be the last entry in the list. The symbolic form may include only read (r), write (w), and execute (x) access. If the acl attribute is defined in the stanza, the SUID, SGID, and SVTX mode attributes are ignored. For a typical mode specification, see the Examples section.

- **owner**  
  Defines the name or numeric ID of the file owner, expressed as the OwnerName or the OwnerID parameter.

- **size**  
  Defines the size of the file in bytes. This attribute is valid only for regular files. The value is a decimal number.

- **type**  
  The type of object. Select one of the following keywords: FILE, DIRECTORY, FIFO, BLK_DEV, CHAR_DEV, or MPX_DE.

Stanzas in this file can be created and altered with the `sysck` command. Direct alteration by other means should be avoided, since other accesses may not be supported in future releases.

Attributes that span multiple lines must be enclosed in double quotes and have new line characters entered as \

Since device configuration and the sysck.cfg database are independent and are not integrated, there is no automatic addition of sysck.cfg entries when a device is added. Hence, given the automatic configuration of devices at boot time, it is the responsibility of the administrator to maintain /etc/security/sysck.cfg. This is also true in the case of mirrored rootvg, since /dev/ipldevice gets relinked dynamically to the other disk when the system is rebooted off the mirrored disk.

**Security**

Access Control: This file should grant read (r) access to the root user and members of the security group, and write (w) access to the root user only. General users do not need read (r) access.

**Examples**

1. A typical stanza looks like the following example for the /etc/passwd file:

   /etc/passwd:
   type = file
   owner = root
   group = passwd
   mode = TCB,640

2. A typical mode specification looks like the following example for a program that is part of the trusted computing base, that is a trusted process, and that has the setuid attribute enabled:
mode = SUID,TP,TCB,rwxr-x---

OR

mode = SUID,TP,TCB,750

Files

/etc/security/sysck.cfg  Specifies the path to the system configuration data base.

Related Information

The `grpck` command, `installp` command, `pwdck` command, `sum` command, `tcbck` command, `usrck` command.

Access control lists in *Operating system and device management*.

Security Administration in *Operating system and device management*.

syslog.conf File

Purpose

Controls output of the `syslogd` daemon.

Description

Each line must consist of two parts:

1. A selector to determine the message priorities to which the line applies.
2. An action. Each line can contain an optional part.
3. Rotation.

The fields must be separated by one or more tabs or spaces.

Format

```
msg_src_list destination [rotate [size sizeklm] [files files] [time timehldlwmy] [compress] [archive archive]]
```

where `msg_src_list` is a semicolon separated list of `facility.priority` where:

- `facility`  all (except mark)
  - mark - time marks kern,user,mail,daemon, auth,... For more information, see *syslogd AIX 5L Version 5.3 Commands Reference, Volume 5*.

- `priority` is one of (from high to low):
  - emerg/panic,alert,crit,err(or),warn(ing),notice,info,debug (meaning all messages of this priority or higher)

- `destination`
  - `/filename` - log to this file
  - `username[],username2...` - write to user(s)
  - `@hostname` - send to syslogd on this machine
  - `*` - send to all logged in users

[rotate [size sizeklm] [files files] [time timehldlwmy] [compress] [archive archive]] is:

If `destination` is a regular file and the word `rotate` is specified, then the `destination` is limited by either `size` or `time`, or both. The `size` value causes the `destination` to be limited to `size`, with `files` files kept in the
rotation. The backup filenames are created by appending a period and a number to destination, starting with .0. The time value causes the destination to be rotated after time. If both time and size are specified, then logfiles will be rotated once the logfile size exceeds size or the after time, whichever is earlier.

If the compress option is specified then the logfile names will be generated with a .Z extension. The files keyword will be applicable to the logfiles which are currently under rotation. For example, if we specify the compress option, then only file with .Z extension will be under rotation and the number of such files will be limited by files files. Any logfiles with an extension other than .Z will not be under the rotation scheme and thus will not be under the restriction of files files. Similarly if the compress option is removed then the files which have been generated with .Z extension will no longer be the part of rotation scheme and will not be limited by the files files.

The minimum size that can be specified is 10k. The minimum number of files that can be specified is 2. The default size is 1MB and the default for files is unlimited. Therefore, if only rotate is specified, the log will be rotated with size = 1m. The compress option means that rotated log files that are not in use will be compressed. The archive option will save rotated log files that are not in use to archive.

The default is not to rotate log files.

Requirement: The letter indicating the unit must immediately follow the number in the syntax. For example, to specify the log rotation of every two days, the phrase time 2d is correct, but time 2 d is not.

Example

"mail messages, at debug or higher, go to Log file. File must exist."
"all facilities, at debug and higher, go to console"
"all facilities, at crit or higher, go to all users"
mail.debug /usr/spool/mqueue/syslog
*.debug /dev/console
*.crit *
*.debug /tmp/syslog.out rotate size 100k files 4
*.crit /tmp/syslog.out rotate time 1d

Files

/etc/syslog.conf Controls the output of syslogd.

Related Information

The syslogd daemon.

targets File

Purpose
Defines iSCSI targets that will be accessed by the iSCSI software initiator.

Description
The iSCSI targets file defines the name and location of the iSCSI targets that the iSCSI software initiator will attempt to access. This file is read any time the iSCSI software initiator driver is loaded.

Any line in this file that begins with "#" will be treated as a comment line and ignored. Other non-blank lines will define a target that the iSCSI software initiator will access. The "\" character may be used between fields as a line continuation character in order to make the file more readable.
The shipped version of this file contains comments that precisely define the format of the file. However, there are no default target names; the shipped file contains only comments. In order to use the iSCSI software initiator, the user must add target definitions to the file and then reload the iSCSI driver by running `cfgmgr` or by rebooting the system.

Each target is defined by three or four fields, as follows:

```
HostNameOrAddress PortNumber iSCSIName
```

or

```
HostNameOrAddress PortNumber iSCSIName CHAPSecret
```

The fields that define the target are:

**HostNameOrAddress**

This is the TCP/IP location of the target. The location may be specified by a TCP/IP address in dotted-decimal form, or by a host name which can be resolved to a valid TCP/IP address. The format for the IP Address is taken from RFC2373.

**PortNumber**

The TCP/IP port number on which the iSCSI target is listening. The standard port number for iSCSI is 3260, but some targets may allow customizing the port number, so this field must be specified.

**iSCSIName**

The iSCSI name of the target. This name must match the name defined to the target. Note that the iSCSI name will be converted to contain all lower case characters, in accordance with the iSCSI standards.

**CHAPSecret**

This optional field specifies the secret to be used by this initiator if CHAP authentication is required. The secret is a text string enclosed in double-quote characters. If this field is included in the target line, the iSCSI software initiator will offer CHAP authentication to the target, and if the target requests such authentication, this value will be used as the secret during the authentication process. If the `CHAPSecret` field is not included in the target definition, the initiator will attempt to log in to the target without any authentication.

**Examples**

1. **iSCSI Target without CHAP(MD5) authentication** - Assume the target is at address 192.168.3.2 and the valid port is 5003. The name of the target is iqn.com.ibm-4125-23wtt26. The target line would look like the following:
   
   ```
   192.168.3.2 5003 iqn.com.ibm-4125-23wtt26
   ```

2. **iSCSI Target with CHAP(MD5) authentication** - Assume the target is at address 10.2.1.105 and the valid port is 3260. The name of the target is iqn.com.ibm-k167-42.fc1a and the CHAP secret is "This is my password." The target line would look like the following:
   
   ```
   10.2.1.105 3260 iqn.com.ibm-k167-42.fc1a "This is my password."
   ```

3. **iSCSI Target with CHAP(MD5) authentication and line continuation** - Assume the target is at address iscsi.fake.com and the valid port is 3260. The name of the target is iqn.2003-01.com.ibm:00.fc0ab21.shark128 and the CHAP secret is "123ismysecretpassword.fc1b". The target line would look like the following:
   
   ```
   iscsi.fake.com 3260 iqn.2003-01.com.ibm:00.fc0ab21.shark128 "123ismysecretpassword.fc1b"
   ```

**Files**

```
/etc/iscsi/targets
```

The iSCSI `targets` file.
Related Information

The iSCSI software initiator in Networks and communication management.

Temporary (TM.*) Files for BNU

Purpose

Store data files during transfers to remote systems.

Description

The Basic Networking Utilities (BNU) temporary (TM.*) files store data files during transfers to remote systems.

After a data (D.*) file is transferred to a remote system by the uucico daemon, the BNU program places the file in a subdirectory of the BNU spooling directory named /var/spool/uucp/SystemName. The SystemName directory is named for the computer transmitting the file. The BNU program creates a temporary data file to hold the original data file.

The full path name of the temporary data file is a form of the following:

/var/spool/uucp/SystemName/TM.xxPID.000

where the SystemName directory is named for the computer sending the file, and TM.xxPID.000 is the name of the file; for example, TM.00451.000. The PID variable is the process ID of the job.

The uucico daemon normally deletes all temporary files when they are no longer needed. However, temporary files can also be removed using the uucleanup command with the -T flag.

Files

/etc/uucp/Systems file
/var/spool/uucp/SystemName directory
/var/spool/uucppublic/* directories
/var/spool/uucp/SystemName/D.* files

Describes accessible remote systems.
Contains BNU command, data, and execute files.
Contain files that BNU has transferred.
Contain data to be transferred.

Related Information

The uucp command, uucleanup command, uudemon.cleanu command, uupick command, uuto command, uux command.

The uucico daemon.

Understanding the BNU Daemons Understanding the BNU File and Directory Structure BNU maintenance commands in Networks and communication management.

Workload Manager .times File

Purpose

Defines time ranges for configurations in a configuration set.
**Description**

Time ranges will appear in the configuration set files. These files are attribute files where the stanzas are the configuration names, and the only attribute is the time range. No default record is allowed (useless and confusing). A missing time range attribute stands for the default time range, which means always outside the other defined time ranges if any.

**Time Coherency Checks**

It is mandatory that the time ranges do not overlap within a single file. In doing so, it would not be possible to find which is the right config to use. The union of all time ranges must cover all times. The default time range will help for ensuring this.

**Note:** It might not be possible to make changes to a correct file that result in another correct file without having intermediate incorrect file contents from this coherency point of view, due to commands or SMIT making one change at a time. For this reason, the content of the file is copied to .running at Workload Manager explicit update time.

**Syntax**

This syntax applies to configuration set files and to `confsetcntrl`, a new command in AIX 5.2, wherever a time range is given. SMIT and Web-based System Manager provide a more convenient way to select or enter a time range. A time range is specified as a range of days with 0 representing Sunday and 6 representing Saturday, and in 24 hour format, with hours and minutes specified. A default time range, which will include all time ranges not otherwise specified, is indicated by a single minus sign (-).

**Specification:**

```
<time-range>: -
<time-range>: <weekday-range>,<time-of-the-day-range>
<time-range>: <weekday-range>
<time-range>: <time-of-the-day-range>
<weekday-range>: <weekday>-<weekday>
<weekday-range>: <weekday>
<weekday>: 0 through 6 for Sunday through Saturday
<time-of-the-day-range>: <time-of-the-day>-<time-of-the-day>
<time-of-the-day>: <hour>,<minute>
<hour>: 0 through 23
<minute>: 0 through 59
```

**Notes:**

1. A colon is accepted to separate hours and minutes instead of dot, provided that the field is quoted (colon has a special meaning in attributes file format).
2. Value 24 is correct for ending <hour> if <minute> is null.
3. For convenience and for command parameters only, <weekday> may be specified with the name or the abbreviation of the day of the week as they appear in the output of `locale day` or `locale abday` commands, taking into account the current user locale (LC_TIME). This is not appropriate for attribute files which do not have a defined locale.

**Example**

```
conf1:     time = -
conf2:     time = "1-5,8:00-17:00"
conf2:     time = "6-0,14:00-17:00"
conf3:     time = "22:00-6:00"
```

**Files**

```
$HOME/.time    Specifies the complete path name of the .time file.
```
Related Information

The `confsetcntrl` and `lswlmconf` commands.

---

**unix.map File**

**Purpose**

Defines the operating system identity used for service provider applications on the node by the UNIX host-based authentication (HBA) security mechanism.

**Description**

Applications that use the cluster security services library must obtain an identity from the security mechanisms supported by the library. These identities are specific to the individual security mechanisms supported by cluster security services. Because cluster security services supports multiple security mechanisms and multiple applications, the cluster security services library must be informed of which identity to use for an application when interacting with a specific security mechanism on its behalf.

The default security mechanism used by the cluster security services library is the HBA mechanism. The `unix.map` file defines the identities used by the core cluster applications when interacting with the HBA mechanism. The cluster security services library expects to locate this file in `/var/ct/cfg/unix.map` (preferred) or `/usr/sbin/rsct/cfg/unix.map` (default).

This file is ASCII-text formatted, and can be modified with a standard text editor. However, this file should not be modified unless the administrator is instructed to do so by the cluster software service provider. If this configuration file is to be modified, the default `/usr/sbin/rsct/cfg/unix.map` file should not be modified directly. Instead, the file should be copied to `/var/ct/cfg/unix.map`, and modifications should be made to this copy. The default configuration file should never be modified.

All entries within this file use the following format:

```
SERVICE:service_name:user_name_running_the_service
```

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>SERVICE</td>
<td>Required keyword</td>
</tr>
<tr>
<td>service_name</td>
<td>Specifies the name commonly used to refer to the application. For example, this could be the name used by the system resource controller to refer to this application.</td>
</tr>
<tr>
<td>user_name_running_the_service</td>
<td>Specifies the operating system user identity used to execute the application process. It is the owner identity that would be seen for the application process in the <code>ps</code> command output.</td>
</tr>
</tbody>
</table>

**Security**

- The default identity mapping definition file `/usr/sbin/rsct/cfg/ctsec_map.global` is readable by all system users, but permissions prevent this file from being modified by any system user.
- When creating the override identity mapping definition files `/var/ct/cfg/ctsec_map.global` and `/var/ct/cfg/ctsec_map.local`, make sure that the files can be read by any system user, but that they can only be modified by the root user or other restrictive user identity not granted to normal system users.
- By default, these files reside in locally-mounted file systems. While it is possible to mount the `/var/ct/cfg` directory on a networked file system, this practice is discouraged. If the `/var/ct/cfg/ctsec_map.local` file were to reside in a networked file system, any node with access to that networked directory would assume that these definitions were specific to that node alone when in reality they would be shared.
Restrictions
This file should not be modified unless the administrator is instructed to do so by the cluster software service provider. Incorrect modification of this file will result in authentication failures for the applications listed in this file and possibly their client applications. If this configuration file is to be modified, the default /usr/sbin/rsct/cfg/unix.map file should not be modified directly. Instead, the file should be copied to /var/ct/cfg/unix.map, and modifications should be made to this copy. The default configuration file should never be modified.

Examples
This example shows the default contents of the configuration file:

```
SERVICE:ctrmc:root
SERVICE:rmc:root
SERVICE:ctloadl:loadl
SERVICE:ctdpcl:root
SERVICE:ctpmd:root
```

Location
/var/ct/cfg/unix.map Contains the unix.map file

Files
/usr/sbin/rsct/cfg/unix.map Default location of the unix.map file

Related Information
Commands: ps
Daemons: ctcasd

updaters File for NIS

Purpose
Updates NIS maps.

Description
The /var/yp/updaters file is a makefile used for updating NIS maps. NIS maps can only be updated in a secure network; that is, one that has a publickey file. Each entry in the file is a make target for a particular NIS map. For example, if there is an NIS map named passwd.byname that can be updated, there should be a make target named passwd.byname in the updaters file with the command to update the file.

The information necessary to make the update is passed to the update command through standard input. All items are followed by a new line except for actual bytes of key and actual bytes of data. The information passed is described below:

- 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 getting this information through standard input, the command to update the map determines whether the user is allowed to make the change. If the user is not allowed, the update command exits with the
YPERR_ACCESS status. If the user is allowed to make the change, the command should make the
change and exit with a status of 0. If any errors exist that may prevent the *updaters* file from making the
change, the command should exit with the status that matches a valid NIS error code described in the
rpcsvc/ypclnt.h file.

### Related Information

- The **publickey** file.
- The **update** command.
- The **ypupdated** daemon.

[Checklist for Administering Secure NFS](#), [Network File System (NFS) Overview for System Management](#), [Network Information Service (NIS) Overview for System Management](#) in *Networks and communication management*.

### user File

#### Purpose

Contains extended user attributes.

#### Description

The `/etc/security/user` file contains extended user attributes. This is an ASCII file that contains attribute
stanzas for users. The **mkuser** command creates a stanza in this file for each new user and initializes its
attributes with the default attributes defined in the `/usr/lib/security/mkuser.default` file.

Each stanza in the `/etc/security/user` file is identified by a user name, followed by a : (colon), and
contains attributes in the form `Attribute=Value`. Each attribute value pair is ended by a new-line character,
and each stanza is ended by an additional new-line character. For an example of a stanza, see the
[Examples](#) section.

The file supports a default stanza. If an attribute is not defined for a user, the default value for the attribute
is used.

#### Attributes

If you have the proper authority, you can set the following user attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>account_locked</td>
<td>Indicates if the user account is locked. Possible values include:</td>
</tr>
<tr>
<td>true</td>
<td>The user's account is locked. The values yes, true, and always are equivalent. The user is denied access to the system.</td>
</tr>
<tr>
<td>false</td>
<td>The user's account is not locked. The values no, false, and never are equivalent. The user is allowed access to the system. This is the default value.</td>
</tr>
<tr>
<td>admin</td>
<td>Defines the administrative status of the user. Possible values are:</td>
</tr>
<tr>
<td>true</td>
<td>The user is an administrator. Only the root user can change the attributes of users defined as administrators.</td>
</tr>
<tr>
<td>false</td>
<td>The user is not an administrator. This is the default value.</td>
</tr>
<tr>
<td>admgroups</td>
<td>Lists the groups the user administers. The Value parameter is a comma-separated list of group names. For additional information on group names, see the <strong>adms</strong> attribute of the <code>/etc/security/group</code> file.</td>
</tr>
<tr>
<td>auditclasses</td>
<td>Lists the user's audit classes. The Value parameter is a list of comma-separated classes, or a value of ALL to indicate all audit classes.</td>
</tr>
</tbody>
</table>


auth1
Lists additional mandatory methods for authenticating the user. The auth1 attribute has been deprecated and may not be supported in a future release. The SYSTEM attribute should be used instead. The authentication process will fail if any of the methods specified by the auth1 attribute fail.

The Value parameter is a comma-separated list of Method;Name pairs. The Method parameter is the name of the authentication method. The Name parameter is the user to authenticate. If you do not specify a Name parameter, the name of the user being authenticated is used.

Valid authentication methods for the auth1 and auth2 attributes are defined in the /etc/security/login.cfg file.

auth2
Lists additional optional methods for authenticating the user. The auth2 attribute has been deprecated and may not be supported in a future release. The SYSTEM attribute should be used instead. The authentication process will not fail if any of the methods specified by the auth2 attribute fail.

The Value parameter is a comma-separated list of Method;Name pairs. The Method parameter is the name of the authentication method. The Name parameter is the user to authenticate. If you do not specify a Name parameter, the name of the user being authenticated is used.

core_compress
Enables or disables core file compression. Valid values for this attribute are On and Off. If this attribute has a value of On, compression is enabled; otherwise, compression is disabled. The default value of this attribute is Off.

core_path
Enables or disables core file path specification. Valid values for this attribute are On and Off. If this attribute has a value of On, core files will be placed in the directory specified by core_pathname (the feature is enabled); otherwise, core files are placed in the user’s current working directory. The default value of this attribute is Off.

core_pathname
Specifies a location to be used to place core files, if the core_path attribute is set to On. If this is not set and core_path is set to On, core files will be placed in the user’s current working directory. This attribute is limited to 256 characters.

core_naming
Selects a choice of core file naming strategies. Valid values for this attribute are On and Off. A value of On enables core file naming in the form core.pid.time, which is the same as what the CORE_NAMING environment variable does. A value of Off uses the default name of core.

daemon
Indicates whether the user specified by the Name parameter can execute programs using the cron daemon or the src (system resource controller) daemon. Possible values are:

true The user can initiate cron and src sessions. This is the default.
false The user cannot initiate cron and src sessions.

dce_export
Allows the DCE registry to overwrite the local user information with the DCE user information during a DCE export operation. Possible values are:

true Local user information will be overwritten.
false Local user information will not be overwritten.

dictionlist
Defines the password dictionaries used by the composition restrictions when checking new passwords.

The password dictionaries are a list of comma-separated, absolute path names that are evaluated from left to right. All dictionary files and directories must be write-protected from all users except root. The dictionary files are formatted one word per line. The word begins in the first column and terminates with a new-line character. Only 7-bit ASCII words are supported for passwords. If text processing is installed on your system, the recommended dictionary file is the /usr/share/dict/words file.

expires
Identifies the expiration date of the account. The Value parameter is a 10-character string in the MMDDhhmmyy form, where MM = month, DD = day, hh = hour, mm = minute, and yy = last 2 digits of the years 1939 through 2038. All characters are numeric. If the Value parameter is 0, the account does not expire. The default is 0. See the date command for more information.
**histexpire**

Designates the period of time (in weeks) that a user cannot reuse a password. The value is a decimal integer string. The default is 0, indicating that no time limit is set.

**histsize**

Designates the number of previous passwords a user cannot reuse. The value is a decimal integer string. The default is 0.

**login**

Indicates whether the user can log in to the system with the **login** command. Possible values are:

- **true**  The user can log in to the system. This is the default.
- **false**  The user cannot log in to the system.

**logintimes**

Specifies the times, days, or both, the user is allowed to access the system. The value is a comma-separated list of entries of the following form:

- `[!]:time-time
  - or-
  - [!]
day[-day][:time-time]
  - or-
  - [!]
date[-date][:time-time]

The **day** variable must be one digit between 0 and 6 that represents one of the days of the week. A 0 (zero) indicates Sunday and a 6 indicates Saturday.

The **time** variable is 24-hour military time (1700 is 5:00 p.m.). Leading zeroes are required. For example, you must enter `0800`, not `800`. The **time** variable must be four characters in length, and there must be a leading colon (:). An entry consisting of only a time specification applies to every day. The start hour of a time value must be less than the end hour.

The **date** variable is a four digit string in the form `mmdd`. `mm` represents the calendar month and `dd` represents the day number. For example, `0001` represents January 1. `dd` may be `00` to indicate the entire month, if the entry is not a range, or indicating the first or last day of the month depending on whether it appears as part of the start or end of a range. For example, `0000` indicates the entire month of January. `0600` indicates the entire month of June. `0311-0500` indicates April 11 through the last day of June.

Entries in this list specify times that a user is allowed or denied access to the system. Entries not preceded by an `!` (exclamation point) allow access and are called ALLOW entries. Entries prefixed with an `!` (exclamation point) deny access to the system and are called DENY entries. The `!` operator applies to only one entry, not the whole restriction list. It must appear at the beginning of each entry.

**loginretries**

Defines the number of unsuccessful login attempts allowed after the last successful login before the system locks the account. The value is a decimal integer string. A zero or negative value indicates that no limit exists. Once the user’s account is locked, the user will not be able to log in until the system administrator resets the user’s `unsuccessful_login_count` attribute in the `/etc/security/lastlog` file to be less than the value of **loginretries**. To do this, enter the following:

```
chsec -f /etc/security/lastlog -s username -a \
  unsuccessful_login_count=0
```

**maxage**

Defines the maximum age (in weeks) of a password. The password must be changed by this time. The value is a decimal integer string. The default is a value of 0, indicating no maximum age.

**maxexpired**

Defines the maximum time (in weeks) beyond the **maxage** value that a user can change an expired password. After this defined time, only an administrative user can change the password. The value is a decimal integer string. The default is `-1`, indicating no restriction is set. If the **maxage** attribute is 0, the password expires when the **maxage** value is met. If the **maxage** attribute is 0, the **maxexpired** attribute is ignored.

**maxrepeats**

Defines the maximum number of times a character can be repeated in a new password. Since a value of 0 is meaningless, the default value of 8 indicates that there is no maximum number. The value is a decimal integer string.

**minage**

Defines the minimum age (in weeks) a password must be before it can be changed. The value is a decimal integer string. The default is a value of 0, indicating no minimum age.
minalpha

Defines the minimum number of alphabetic characters that must be in a new password. The value is a decimal integer string. The default is a value of 0, indicating no minimum number.

mindiff

Defines the minimum number of characters required in a new password that were not in the old password. The value is a decimal integer string. The default is a value of 0, indicating no minimum number.

minlen

Defines the minimum length of a password. The value is a decimal integer string. The default is a value of 0, indicating no minimum length. The maximum value allowed is 8. This attribute is determined by the minalpha attribute value added to the minother attribute value. If the sum of these values is greater than the minlen attribute value, the minimum length is set to the result.

minother

Defines the minimum number of non-alphabetic characters that must be in a new password. The value is a decimal integer string. The default is a value of 0, indicating no minimum number.

projects

Defines the list of projects that the user’s processes can be assigned to. The value is a list of comma-separated project names and is evaluated from left to right. The project name should be a valid project name as defined in the system. If an invalid project name is found on the list, it will be reported as an error by the user command.

pwdchecks

Defines the password restriction methods enforced on new passwords. The value is a list of comma-separated method names and is evaluated from left to right. A method name is either an absolute path name or a path name relative to /usr/lib of an executable load module.

pwdwarntime

Defines the number of days before the system issues a warning that a password change is required. The value is a decimal integer string. A zero or negative value indicates that no message is issued. The value must be less than the difference of the maxage and minage attributes. Values greater than this difference are ignored, and a message is issued when the minage value is reached.

registry

Defines the authentication registry where the user is administered. It is used to resolve a remotely administered user to the local administered domain. This situation may occur when network services unexpectedly fail or network databases are replicated locally. Example values are files or NIS or DCE.

rlogin

Permits access to the account from a remote location with the telnet or rlogin commands. Possible values are:

true The user account can be accessed remotely. This is the default rlogin value.

false The user account cannot be accessed remotely.

su

Indicates whether another user can switch to the specified user account with the su command. Possible values are:

true Another user can switch to the specified account. This is the default.

false Another user cannot switch to the specified account.

sugroups

Lists the groups that can use the su command to switch to the specified user account. The Value parameter is a comma-separated list of group names, or a value of ALL to indicate all groups. An ! (exclamation point) in front of a group name excludes that group. If this attribute is not specified, all groups can switch to this user account with the su command.
SYSTEM

Defines the system authentication mechanism for the user. The value may be an expression describing which authentication methods are to be used or it may be the keyword NONE.

The SYSTEM mechanism is always used to authenticate the user, regardless of the value of the auth1 and auth2 attributes. If the SYSTEM attribute is set to NONE, authentication is only performed using the auth1 and auth2 attributes. If the auth1 and auth2 attributes are blank or ignored, as with the TCP socket daemons (ftpd, rexec and rshd), no authentication will be performed.

The method names compat, files and NIS are provided by the security library. Additional methods may be defined in the file /usr/lib/security/methods.cfg.

Specify the value for SYSTEM using the following grammar:

"SYSTEM" ::= EXPRESSION
EXPRESSION ::= PRIMITIVE | "("EXPRESSION")" | EXPRESSION OPERATOR EXPRESSION
PRIMITIVE ::= METHOD | METHOD "["RESULT"]" | "UNAVAIL" | "*"
RESULT ::= "SUCCESS" | "FAILURE" | "NOTFOUND" | "UNAVAIL" | "*
OPERATOR ::= "AND" | "OR"
METHOD ::= "compat" | "files" | "NONE" | [a-z,A-Z,0-9]*

An example of the syntax is:
SYSTEM = "DCE OR DCE[UNAVAIL] AND compat"

tpath

Indicates the user's trusted path status. The possible values are:

always The user can only execute trusted processes. This implies that the user's initial program is in the trusted shell or some other trusted process.
notsh The user cannot invoke the trusted shell on a trusted path. If the user enters the secure attention key (SAK) after logging in, the login session ends.
nosak The secure attention key (SAK) is disabled for all processes run by the user. Use this value if the user transfers binary data that may contain the SAK sequence. This is the default value.
on The user has normal trusted path characteristics and can invoke a trusted path (enter a trusted shell) with the secure attention key (SAK).

ttys

Lists the terminals that can access the account specified by the Name parameter. The Value parameter is a comma-separated list of full path names, or a value of ALL to indicate all terminals. The values of RSH and REXEC also can be used as terminal names. An ! (exclamation point) in front of a terminal name excludes that terminal. If this attribute is not specified, all terminals can access the user account. If the Value parameter is not ALL, then /dev/pts must be specified for network logins to work.

umask

Determines file permissions. This value, along with the permissions of the creating process, determines a file's permissions when the file is created. The default is 022.

Changing the user File

You should access this file through the commands and subroutines defined for this purpose. You can use the following commands to change the user file:

- chuser
- Isuser
- mkuser
- rmuser
The `mkuser` command creates an entry for each new user in the `/etc/security/user` file and initializes its attributes with the attributes defined in the `/usr/lib/security/mkuser.default` file. To change attribute values, use the `chuser` command. To display the attributes and their values, use the `lsuser` command. To remove a user, use the `rmuser` command.

To write programs that affect attributes in the `/etc/security/user` file, use the subroutines listed in `Related Information`.

**Security**

**Access Control**
This file should grant read (r) access only to the root user and members of the security group. Access for other users and groups depends upon the security policy for the system. Only the root user should have write (w) access.

**Auditing Events:**

<table>
<thead>
<tr>
<th>Event</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>S_USER_WRITE</td>
<td>file name</td>
</tr>
</tbody>
</table>

**Examples**

1. A typical stanza looks like the following example for user `dhs`:

   ```
   dhs:
   login = true
   rlogin = false
   ttys = /dev/console
   sugroups = security,!staff
   expires = 0531010090
   tpath = on
   admin = true
   auth1 = SYSTEM,METH2;dhs
   ```

2. To allow all ttys except `/dev/tty0` to access the user account, change the `ttys` entry so that it reads as follows:

   ```
   ttys = !/dev/tty0,ALL
   ```

**Files**

- `/etc/group`
- `/etc/passwd`
- `/etc/security/audit/config`
- `/etc/security/environ`
- `/etc/security/group`
- `/etc/security/limits`
- `/etc/security/login.cfg`
- `/etc/security/password`
- `/usr/lib/security/mkuser.default`
- `/etc/security/user`
- `/etc/security/lastlog`

Contains the basic group attributes.

Contains the basic user attributes.

Contains audit system configuration information.

Contains the environment attributes of users.

Contains the extended attributes of groups.

Contains the process resource limits of users.

Contains configuration information for user log in and authentication.

Contains password information.

Contains default user configurations.

Contains extended user attributes.

Contains last login information.

**Related Information**

The `chuser` command, `lsuser` command, `mkuser` command, `rmuser` command.

The `enduserdb` subroutine, `getuserattr` subroutine, `putuserattr` subroutine, `setuserdb` subroutine.
For more information about the identification and authentication of users, discretionary access control, the trusted computing base, and auditing, refer to Security Administration and Users, roles, and passwords in Security.

user.roles File

Purpose
Contains the list of roles for each user. This system file only applies to AIX 4.2.1 and later.

Description
The /etc/security/user.roles file contains the list of roles for each user. This is an ASCII file that contains a stanza for system users. Each stanza is identified by a user name followed by a : (colon) and contains attributes in the form Attribute=Value. Each attribute pair ends with a newline character as does each stanza.

This file supports a default stanza. If an attribute is not defined, either the default stanza or the default value for the attribute is used.

A stanza contains the following attribute:

roles Contains the list of roles for each user.

For a typical stanza, see the Examples section.

Typically, the /etc/security/user.roles stanza contains an entry for every user and a list of data associated with that user. The roles database does not require an entry per user. The size of each entry is one line.

The user.roles file is kept separately from the /etc/security/user file for performance reasons. Several commands scan this database, so system performance increases with smaller files to scan (especially on systems with large numbers of users).

Changing the user.roles File
You should access this file through the commands and subroutines defined for this purpose. You can use the following commands to change the user.roles file:

- chuser
- Isuser
- mkuser

The mkuser command creates an entry in the /etc/security/user.roles file for each new user when the roles attribute is used. To change the attribute values, use the chuser command with the roles attribute. To display the attributes and their values, use the Isuser command with the roles attribute.

To write programs that affect attributes in the /etc/security/user.roles file, use the subroutines listed in Related Information.

Security
Access Control: This file grants read and write access to the root user, and read access to members of the security group.

Examples
A typical stanza looks like the following example for the username role:

```
username:roles=role1,role2,role3
```
username:
  roles = role1,role2

Files

/etc/security/roles
Contains the list of valid roles.
/etc/security/user.roles
Contains the list of roles for each user.
/etc/security/smitacl.group
Contains the group ACL definitions.
/etc/security/smitacl.user
Contains the user ACL definitions.

Related Information
The `chuser` command, `lsuser` command, `mkuser` command.
The `getuserattr` subroutine, `putuserattr` subroutine.

vfs File

Purpose
Describes the virtual file systems (VFS) installed on the system.

Description
The `/etc/vfs` file describes the virtual file systems installed on the system. The name, type number, and file-system helper program are among the types of information listed in the file. Commands, such as the `mount` command, the `fsck` command (file system check), and the `mkfs` command (make file system), use this information.

The `vfs` file is an ASCII file, with one record per line. The following are examples of the three types of lines in the `vfs` file:

- Comments
  # This is a comment.
  # Comments begin with a # (pound sign).
  # Blank lines are ignored.
  # The following example only locally defines the default `vfs` file.

- General control
  `%defaultvfs jfs nfs`
  The fields for the `%defaultvfs` control line are:

  `%defaultvfs` Identifies the control line.
  jfs Indicates the default local virtual file system.
  nfs Indicates the remote virtual file system (optional).

- Entries
  #Name Type Mount Helper Fs. helper
  jfs 3 none /sbin/helpers/v3fshelper
  nfs 2 /etc/nfsnethelp none
  cdrfs 5 none none

The comments are in text for explanatory purposes. The general control lines, which are designated by a `%` (percent) character, configure the actions of the following commands:

- `mount`
- `umount`
- `mkfs`
For example, a line like `%defaultvfs` indicates the default local virtual file system is used if no VFS is specified by the `mount` command or in the `/etc/filesystems` file. The entry is the name of the VFS as indicated in the file. If a second entry is listed on the same line, it is taken to be the default remote VFS. The `%defaultvfs` control line may leave off the remote VFS specification.

The VFS entries take the following form:

- **name**: Canonical name of this type of virtual file system.
- **type**: Decimal representation of the virtual file system type number for the VFS.
- **mnt_helper**: Path name of the mount helper program of this VFS. If a mount helper is not required, the entry should be displayed as `none`. If this path name does not begin with a slash, it is relative to the `/sbin/helpers` directory.
- **fs_helper**: Path name of the file system helper program of this VFS. If a file system helper is not required, the entry should be `none`. If this path name does not begin with a slash, it is relative to the `/sbin/helpers` directory.

### Files

- **/etc/filesystems**: Lists the known file systems and defines their characteristics.

### Related Information

The commands `chvfs` command, `crvfs` command, `df` command, `ff` command, `fsck` command, `fsdb` command, `lsvfs` command, `mkfs` command, `mount` command, `rmvfs` command, `umount` command.

The File systems in Operating system and device management explains file system types, management, structure, and maintenance.

## Workload Manager classes File

### Purpose

Contains the definition of Workload Manager (WLM) superclasses or subclasses for a given configuration.

### Description

The **classes** file in the `/etc/wlm/Config` directory describes the superclasses of the WLM configuration, `Config`. If the superclass `Super` of this configuration has subclasses defined, these subclasses are defined in the file `/etc/wlm/Config/Super/classes`.

Some attributes apply to only superclasses or to only subclasses. The description of the **classes** file uses the terms **class** or **classes** when a statement applies to both superclasses and subclasses.

The **classes** file is organized into stanzas. Each stanza names a WLM class and contains attribute-value pairs that describe characteristics of the class.

### Attributes

Each stanza names a WLM class. Class names can contain only upper- and lowercase letters, numbers, and underscores. They are limited to 16 characters in length. The only names that have special meaning
to the system are Default, Shared, Unclassified, Unmanaged, and System. You cannot use Unclassified and Unmanaged as class names. The superclasses Default, Shared, and System are always defined. The subclasses Default and Shared are always defined.

The following attributes are defined in the **classes** file:

- **tier**: Specifies the position of the class in the hierarchy of resource limitation desirability for all classes. A class with a lower tier value will be more favored than a class with a higher tier value. The tier value is a number from 0 to 9. If this attribute is not defined, it defaults to 0.

- **inheritance**: If the `inheritance` attribute is given the value Yes, the children of processes in this class remain in the class upon execution, regardless of the automatic assignment rules in effect. If this attribute is given No, the normal assignment rules apply. If not defined, the attribute defaults to No.

- **localshm**: Indicates whether memory segments accessed by processes in different classes remain local to the class they were initially assigned to or if they go to the Shared class. The possible value is Yes or No. If not specified, the default is No.

- **authuser**: Specifies the user name of the user allowed to assign processes to this class. If not defined, this attribute defaults to the empty string (""").

- **authgroup**: Specifies the group name of the group of users allowed to assign processes to this class. If not defined, the attribute defaults to the empty string.

- **rset**: Names the resource set to which the processes in the class have access. If the attribute is not defined, it defaults to an empty string, meaning that the class has access to all the resources on the system.

- **adminuser**: Specifies the user name of the user allowed to administer the subclasses of this superclass. If not defined, the attribute defaults to the empty string.

This attribute is valid only for superclasses.

- **admingroup**: Specifies the group name of the group of users allowed to administer the subclasses of this superclass. (Primary group of users should match with this group name.) If this attribute is not defined, it defaults to the empty string.

This attribute is valid only for superclasses.

- **delshm**: If set to "yes", or if a killed process due to a virtual memory limit is the last process referencing a shared segment, the segment is deleted. The default is to not delete the shared segments (value set to "no").

- **vmenforce**: When a class reaches its virtual memory limit, if `vmenforce` is set to "class", all of the processes classified to the faulting class are killed. If `vmenforce` is set to "proc" (default), then only the process that pushes the usage past the virtual memory limit is killed.

The attributes that have not been explicitly set by a WLM administrator using any of the administration tools (file editing, command line, or SMIT) are omitted in the property files.

The default values mentioned above are the system defaults and can be modified using a special stanza named "default."

**Files**

- **classes**: Defines the superclasses or subclasses of a WLM configuration

**Security**

The WLM property files defining the superclasses of a WLM configuration must have write permission only for the root user. The WLM property files defining the subclasses of a superclass must have write permission for the adminuser and admingroup for the superclass.
If there is no adminuser for the superclass, the files should be owned by root. If no admingroup exists for a superclass, the WLM property files for the superclass should be group "system" with no write permission for group.

**Example**

1. The following entry at the beginning of the **classes** file modifies the default values for the **tier** and **inheritance** attributes so that if they are not defined for some (or all) of the classes specified in the file, the **tier** value is 1 and the **inheritance** value is Yes:

   ```
   default:
   tier = 1
   inheritance = "yes"
   ```

   The scope of these user-defined default values is limited to the file where they appear. For instance, if the above default stanza appears in the top-level **classes** file of a configuration, it does not affect the default values for the **classes** files defining the subclasses of the various superclasses.

2. The following is an example of a typical **/etc/wlm/Config/classes** file:

   ```
   * system defined classes
   * All attributes to default value
   * Attribute values can be specified
   *
   Default:
   System:
   Shared:
   * User defined classes
   *
   Super1:
   inheritance = "yes"
   adminuser = "bob"
   authgroup = "devlt"
   Super2:
   tier = 4
   localshm = "yes"
   admingroup = "sales"
   authuser = "sally"
   rset = "part1"
   ```

   **Note:** The asterisk (*) is a comment character. Comments are added by directly editing the file. However, when you use the command line, or SMIT to create, modify, or delete classes, the comments are removed.

**Related Information**

The **lsclass** command, **mkclass** command, **chclass** **rmclass** command.

The **shares** file, **limits** file, **rules** file.

[Workload management] in *Operating system and device management.*

---

**Workload Manager limits File**

**Purpose**

Describes the minimum and maximum limits for the resources allocated to superclasses or subclasses of a WLM configuration.

**Description**

The **limits** file in the **/etc/wlm/Config** describes the resource limits for the superclasses of the WLM configuration **Config**. If the superclass **Super** of this configuration has subclasses defined, the resource limits for the subclasses are defined in the file **/etc/wlm/Config/Super/limits**.
The limits at the superclass level represent a percentage of the total amount of resources available on the system and the limits at the subclass level represent a percentage of the resource made available to the parent superclass. Despite this difference, the description of resource limits is relevant to both a superclass and a subclass.

The **limits** file is organized into stanzas that are named after WLM classes and contain attribute-value pairs specifying the minimum and maximum resource limit allocated to the class for the various resources. The attribute names identify the resource. For each resource, the following values must be provided:

- Minimum limit (expressed here as \( m \))
- Soft maximum limit (expressed here as \( SM \))
- Hard maximum limit (expressed here as \( HM \))

The limits are expressed as percentages. Both the minimum and maximum limits are each a number between 0 and 100. The hard maximum must be greater than or equal to the soft maximum, which in turn must be greater than or equal to the minimum. When the limits are not specified for a class or a resource type, the system defaults to 0 for the minimum and 100 for both the soft and hard maximum.

Use the following format for defining the limit values:

```
attribute_name = m%-SM, HM%
```

In AIX 5.2 and later, you can also specify per-process and per-class total limits. These are hard limits and can be specified in the following format:

```
attribute_name = <value> [unit]
```

The valid range of values for each attribute, as well as their default and allowed units are described in *Operating system and device management*.

**Attributes**

Each stanza names a WLM class that must exist in the **classes** file at the corresponding level (superclass or subclass).

The following attributes are defined in the **limits** file:

- **CPU**
  Represents the CPU limits for the class

- **memory**
  Represents the physical memory limits for the class

- **diskIO**
  Represents the disk I/O limits for the class

- **totalConnectTime**
  The maximum amount of time a login session in the class can stay active. This is specified as an integer with the units intended (\( s \) for seconds, \( m \) for minutes, \( h \) for hours, \( d \) for days, and \( w \) for weeks). As a user approaches this connection time limit, WLM will send warning messages. When the limit is reached, the user will be notified and the login session will be terminated.

- **totalCPU**
  The total amount of CPU time allowed for each process in the class. This is specified as an integer with the units intended (\( s \) for seconds, \( m \) for minutes, \( h \) for hours, \( d \) for days, and \( w \) for weeks).

- **totalDiskIO**
  The total amount of DiskIO allowed for each process in the class. This is specified as an integer with the units intended (\( KB \) for kilobytes, \( MB \) for megabytes, \( TB \) for terabytes, \( PB \) for petabytes, and \( EB \) for exabytes).

- **totalLogins**
  The total number of login sessions simultaneously available in the class. If a user tries to log onto the system and the login shell would end up in a class that has reached the **totalLogins** limit, the login operation will fail. Also, if an operation would cause a login shell to be moved into a class that has reached the **totalLogins** limit, the operation will also fail.

- **totalProcesses**
  The maximum number of processes allowed in the class. If an operation would result in a new process entering the class when the class has this many processes in it, the operation will fail.
**totalThreads**

The maximum number of threads allowed in the class. If an operation would result in a new thread entering the class when the class has this many threads in it, the operation will fail. The total thread limit must be at least as large as the total process limit for a class. If a class has a total thread limit but no total process limit specified, the total process limit will be set to the total thread limit.

**classVirtMem**

The maximum amount of virtual memory that a class can have at one time. This is specified as an integer with units intended (MB for megabytes, GB for gigabytes, TB for terabytes).

**procVirtMem**

The maximum amount of virtual memory that a process can have at one time. This is specified as an integer with units intended (MB for megabytes, GB for gigabytes, TB for terabytes).

The default values mentioned above are the system defaults and can be modified using a special stanza named "default."

Consider the following stanza at the beginning of the `limits` file:

```
default:
  CPU = 10%-50%,80%
  diskIO = 20%-60%,100%
```

This stanza modifies the default values of the limits for CPU and disk I/O so that if those attributes are not specified for some (or all) of the classes, their minimum, soft maximum, and hard maximum default to the values shown above. In this example, the default values for the physical memory limits (specified by the `memory` attribute) are still the system default—the minimum 0% and the soft and hard maximum each 100%.

Classes that use only default values for all the resource types can be omitted in the file.

**Security**

The `limits` file defining the limits of the superclasses of a WLM configuration must have write permission for root only. The `limits` file defining the limits for the subclasses of a superclass must have write permission for the adminuser and admingroup for the superclass. If no adminuser exists for the superclass, the `limits` file should be owned by root. If no admingroup exists for a superclass, the file for the superclass should be owned by the "system" group and have no write permission for group.

**Example**

The following is an example of a typical `/etc/wlm/Config/limits` file:

```
* System Defined Classes
* In this example, the system administrator uses
* only default values for the System and Shared
* superclasses. The System class has a memory minimum of
* 1% by default - can be increased by system administrator
* The system administrator gives non default values
* only for the Default class:
* 
  System:
     memory = 1%-100%,100%

Default:
  CPU = 0%-50%,75%
  memory = 0%-25%,50%

* 
* User defined classes
* 
  Super1:
     CPU = 10%-100%,100%
     memory = 20%-100%,100%
```
diskIO = 0%-33%,50%
Super2:
  memory =0%-20%,50%
  diskIO =10%-66%,100%

Note: The asterisk (*) is a comment character.

Files
limits

Related Information
The \texttt{lsclass} command, \texttt{mkclass} command, \texttt{chclass} command, \texttt{rmclass} command.

The \texttt{classes} file, \texttt{shares} file, \texttt{rules} file.

Files in \textit{Operating system and device management}.

Workload management in \textit{Operating system and device management}.

\section*{Workload Manager rules File}

\subsection*{Purpose}
Defines the automatic class assignment rules for the superclasses or subclasses of a given Workload Manager (WLM) configuration.

\subsection*{Description}
The \texttt{/etc/wlm/Config/rules} file describes the assignment rules for the superclasses of the WLM configuration \textit{Config}. If the superclass \textit{Super} of this configuration has subclasses defined, the assignment rules for the subclasses are defined in the file \texttt{/etc/wlm/Config/\textit{Super}/rules}.

The assignment rules for the superclasses and subclasses are formatted the same.

The \texttt{rules} file is a standard text file. Each line represents an assignment rule for a specified class. Several assignment rules can exist at the same time for the same class. Each rule lists the name of a class and a list of values for some attributes of a process; these values are used as classification criteria. The various fields of a rule are separated by white spaces. For attributes that you do not define, represent them with a hyphen (-).

Note: Implicit rules exist for the default superclass and the default subclass of every superclass. These rules catch all processes that did not match any rules explicitly stated in the \texttt{rules} file. They apply to a WLM configuration regardless of whether they are explicitly stated.

Assignment Rules
Assignment rules are made of the fields described below. When present, the following fields \textit{must} appear in the order shown below. Order is important because the values are given to the field identified by its position in the string. Only the first three fields (\texttt{class}, \texttt{reserved}, and \texttt{user}) are mandatory. The remaining fields can be omitted if their values are hyphens.

For instance, the assignment rule
\texttt{class1 - user1}

is the same as the following:
The rule
class - - group1 - 32bit+fixed

is a valid rule equivalent to the following rule:

class - - group1 - 32bit+fixed -

But in the case of the rule class1 - group1, group1 is interpreted as a user name because of its position in the string.

class
Contains the name of a class that is defined in the class file corresponding to the level of the rules file (superclass or subclass). Class names can contain only upper- and lowercase letters, numbers, and underscores and are limited to 16 characters in length.

No assignment rule can be specified for the system-defined classes "Unclassified," "Unmanaged," and "Shared."

reserved
This field must be set to a hyphen (-)

user
Contains either a hyphen or a list of valid user names (as defined in the /etc/passwd file). The list is composed of one or more names, separated by a comma (,). To exclude a user from the class, place an exclamation mark (!) before the name of that user. Use full Korn shell pattern-matching syntax to specify a wildcard pattern to match a set of user names.

group
Contains either a hyphen or a list of valid group names (as defined in the /etc/group file). The list is composed of one or more names, separated by a comma. To exclude a group from the class, place an exclamation mark before the name of that group. Use full Korn shell pattern-matching syntax to specify a wildcard pattern to match a set of group names.

application
Contains either a hyphen or a list of application path names. This is the path name of the file executed by the processes to be included in the class. The value can be either a full path name or a wildcard pattern that matches a set of path names. The list is composed of one or more path names, separated by a comma. To exclude an application from the class, place an exclamation mark before the name of that application.

type
Contains either a hyphen or a list of attributes for the process. The following are possible values:

32bit Indicates that the process is 32-bit
64bit Indicates that the process is 64-bit
plock Indicates that the process has called the plock subroutine to pin memory
fixed Indicates that the process is fixed priority (SCHED_FIFO or SCHED_RR)

The value of the type field can be a comma-separated list of combinations of one or more of the values above. Within the combination, each item must be separated by a plus (+) sign. For example, the value fixed,64bit+plock indicates that any fixed priority process (whether 32- or 64-bit) matches. In addition, 64-bit processes calling the plock subroutine matches. But the value fixed+64bit+plock indicates a different rule criteria: only processes that are 64-bit, fixed, and that are calling the plock subroutine match the criteria.

The 32bit value and 64bit value mutually exclude each other.

tag
May contain either a hyphen or a list of application tags. The list is composed of one or more application tag values separated by commas.

When classifying a process, WLM attempts to match the values of the process attributes (user, group, application, type, and tag) with the values provided in the rules file. To match values, WLM uses the following criteria:

- If the value in the rule is a hyphen, any value of the corresponding process attribute is a match.
- If the value of a process attribute (other than the type attribute) appears in the list of values specified in the corresponding field in the rule and is not preceded by an exclamation mark, it is a match for the specified attribute.
• If the values of the process type attribute (32bit/64bit, plock, fixed) match all the values (separated by + signs) provided in the list for the type field in the rule, they are a match for the process type.
• The process is classified in the class specified in the class field of the rule if ALL the values of the process attributes match the values in the corresponding field of the rule.
• WLM scans the rules in the order in which they appear in the rules file and classifies the process in the class specified in the first rule for which a match is detected. Therefore, the order of the rules in the file is very important.

When classifying a process, WLM first scans the rules file for the superclasses of the current configuration to determine which superclass the process is assigned to. Then, WLM scans the rules file for this superclass to determine which subclass it assigns the process to.

Threads may also be assigned to a class if they have an application tag. They will be assigned using the same rules as the process. However, threads that do not have an application tag will remain assigned to their process' class.

Groupings
As an improvement for AIX 5.2, you can now use attribute value groupings in the rules file. Attribute groupings can be used as element of a selection criteria in the rules file for superclasses or subclasses. For more information, syntax and examples, see the groupings file.

Security
The file containing the assignment rules for the superclasses must have write permission for the root user only. The file containing the assignment rules for the subclasses of a superclass must have write permission for the adminuser and admingroup for the superclass. If no adminuser exists for the superclass, the file should be owned by root. If no admingroup exists for a superclass, the rules file for the superclass should be owned by the system group and have no write permission for group.

Examples
1. The following is an example of a /etc/wlm/Config/rules file:

   * This file contains the rules used by WLM to
   * assign a process to a superclass
   *
   * class resvd user group application type tag
db1 - - - /usr/bin/oracle* - _DB1
   db2 - - - /usr/bin/oracle* - _DB2
devil - - dev - - -

   VPs - bob,ted - - - -
acctg - - acct* - - -
System - root - - - -
Default - - - - - -

2. The following is an example of the rules file for the superclass devil in the /etc/wlm/Config/devlt/ directory:

   * This file contains the rules used by WLM to
   * assign a process to a subclass of the
   * superclass devil
   *
   * class resvd user group application type tag
hackers - jim,liz - - - -
hogs - - - - - 64bit+plock -
editors - !sue - /bin/vi,/bin/emacs - -
buid - - - /bin/make,/bin/cc - -
Default - - - - - -

Note: The asterisk (+) is a comment character.
Files
rules

 Defines the class assignment rules for the superclasses or subclasses of a WLM configuration

Related Information
The wlmcheck command.
The classes file, limits file, shares file.

Files

Workload management in Operating system and device management.

Workload Manager shares File

Purpose
Contains the definition of the number of shares of all the resources allocated to superclasses or subclasses for a given configuration.

Description
The shares file in the /etc/wlm/Config directory describes the resource allocations for the superclasses of the WLM configuration named Config. If the superclass named Super of this configuration has subclasses defined, the resource allocations for the subclasses are defined in the file /etc/wlm/Config/Super/shares.

The file is organized into stanzas that are named after WLM classes and contain attribute-value pairs specifying the number of shares allocated to the class for the various resources. The attribute names identify the resource. The shares value is either an integer between 1 and 65535 or a hyphen (-) to indicate that WLM does not regulate the class for the given resource. The hyphen is the system default.

Attributes
Each stanza names a WLM class that must exist in the classes file at the corresponding level (superclass or subclass).

The following are class attributes defined in the shares file:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Specifies the number of CPU shares allocated to the class</td>
</tr>
<tr>
<td>memory</td>
<td>Specifies the number of physical memory shares allocated to the class</td>
</tr>
<tr>
<td>diskI0</td>
<td>Specifies the number of disk I/O shares allocated to the class</td>
</tr>
</tbody>
</table>

The default values mentioned above are the system default and can be modified using a special stanza named “default.”

Consider the following stanza at the beginning of the shares file:

default:
    CPU     = 10
    diskI0  = 4

This stanza defines the default values for the number of shares for CPU and disk I/O so that if the CPU and diskI0 attributes are not specified for some or all of the classes specified, the attributes default to 10 and 4, respectively. In this example, the default value for physical memory is still a hyphen, meaning no regulation. Classes that use only default values for all the resource types can be omitted in the shares file.
Security

The shares file must have write permission for root user only. The shares file for superclasses must have write permission for the adminuser and admingroup for the superclass.

If no adminuser exists for the superclass, the files should be owned by root. If no admingroup exists for a superclass, the shares file for the superclass should be owned by the system group and should have no write permission for group.

Example

The following is an example of a typical /etc/wlm/Config/shares file:

* System Defined Classes
  * In this example, the system administrator uses
  * only default values for the System
  * and Shared
  * superclasses, and those are omitted
  * in the file
  * The system administrator gives non
  * default values
  * only for the Default class:

* Default:
  * CPU = 5
  * memory = 10

* User defined classes

* Super1:
  * CPU = 8
  * memory = 20
  * diskIO = 6

  Super2:
  * memory = 12
  * diskIO = 6

Note: The asterisk (*) is a comment character.

Files

shares Defines the resource entitlements for the superclasses or subclasses of a WLM configuration

Related Information

The lsclass command, mkclass command, chclass command, rmclass command.

The classes file, limits file, rules file.

Workload management in Operating system and device management.

xferstats File for BNU

Purpose

Contains information about the status of file transfer requests.
Description
The /var/spool/uucp/.Admin/xferstats file contains information about the status of each Basic Networking Utilities (BNU) file transfer request. The xferstats file contains:
- System name
- Name of the user requesting the transfer
- Date and time of the transfer
- Name of the device used in the transfer
- Size of the transferred file
- Length of time the transfer took

Examples
Following is a typical entry in the xferstats file:

zeus!jim M (10/11-16:10:33) (C,9234,1) [-] -> 1167 / 0.100secs

A file was transferred by user jim to system zeus at 4:10 p.m. on the 11th of October. The file size was 1167 bytes and the transfer took 0.100 seconds to complete.

Files
/var/spool/uucp/.Admin directory Contains the xferstats file and other BNU administrative files.

Related Information
The uucp command, uudemon.cleanu command, uux command.
The cron daemon, uucico daemon, uuxqt daemon.
Understanding the BNU File and Directory Structure and BNU maintenance in Networks and communication management.

xtab File for NFS

Purpose
Contains entries for currently exported NFS directories.

Description
The /etc/xtab file contains entries for directories that are currently exported. This file should only be accessed by programs using the getexportent subroutine. To remove entries from this file, use the -u flag of the exportfs command.

Files
/etc/exports Lists the directories that the server can export.
/etc/hosts Contains an entry for each host on the network.
/etc/netgroup Contains information about each user group on the network.
Related Information

The `exportfs` command.

NFS Services in *Networks and communication management.*
Chapter 2. File Formats

Certain files in the operating system are required to have a specific format. The formats of the files that are provided with the operating system are discussed in the documentation for those files. If a file is generated by either the system or a user rather than provided on the distribution medium, it is discussed as a *file format* in this documentation. File formats often are also associated with header files that contain C-language definitions and structures for the files.

More information about the following file formats is provided in this documentation:

- **acct**: Describes the format of the records in the system accounting files.
- **ar**: Describes the format of an archive file.
- **audit**: Describes values used by the auditing system as well as the structure of a bin.
- **bootptab**: Describes the default configuration database for the Internet Boot Protocol server (*bootpd*).
- **CGRM**: Defines a file format for storage and retrieval of device-independent graphics.
- **charmap**: Defines character symbols as character encodings.
- **core**: Describes the structures created in a *core* file as a result of a core dump.
- **cpio**: Describes the *cpio* (copy in/out) archive file.
- **eqnchar**: Contains special character definitions for the *eqn* and *neqn* commands.
- **lastlog**: Defines the last login attributes for users.
- **ldapattribmap**: Defines AIX to LDAP attribute name mapping to support configurable LDAP server schema.
- **locale definition**: Contains one or more categories that describe a locale.
- **locale method**: Specifies the methods to be overridden when constructing a locale.
- **magic**: Defines file types.
- **mailrc**: Sets defaults for the *mail* command.
- **mh_alias**: Defines aliases for the Message Handler (MH).
- **mib_defs**: Provides descriptions of Management Information Base (MIB) variables for the *snmpinfo* command.
- **nroff** and **troff** input
- **nterm**: Describes the format of the terminal driver tables for the *nroff* command.
- **profile**: Describes the format of the *profile* and *.profile* files, which set the user environment at login time.
- **queuedefs**: Specifies input file format for the *nroff* and *troff* commands.
- **sccsfile**: Describes the format of the terminal driver tables for the *nroff* command.
- **setmaps**: Describes the format of the *profile* and *.profile* files, which set the user environment at login time.
- **terminfo**: Defines the text of a code-set map file and a terminal map file.
- **TIFF**: Enables InfoCrafter to support scanned images that have been imported into Interleaf documents.
- **trcfmt**: Stores trace templates.
- **troff**: Describes the output language of the *troff* command.
- **troff Font**: Describes the format of the *troff* command font files.
- **tunables**: Centralizes tunable parameter values.
- **UIL**: Contains information on the user interface for a widget-based application.
- **utmp**, **wtmp**, **failedlogin**: Describes the format of the user and accounting information in the *utmp*, *wtmp*, and *failedlogin* files.
- **vgrinddefs**: Contains the language definition database for the *vgrind* command.
- **WML**: Generates variable UIL compiler components.
Asynchronous Terminal Emulation (ATE) File Formats

- **ate.def**: Determines default settings for use in asynchronous connections and file transfers.
- **ATE Dialing Directory**: Lists phone numbers that the ATE program uses to establish modem connections.

Basic Networking Utilities (BNU) File Formats

- **Devices**: Contains information about devices on the local system that can establish a connection to a remote computer using the Basic Networking Utilities (BNU) program.
- **Dialcodes**: Contains the initial digits of telephone numbers used to establish remote connections over a phone line.
- **Dialers**: Lists modems used for Basic Networking Utilities (BNU) remote communications links.
- **Maxuuscheds**: Limits the number of instances of the `uusched` and `uucico` daemons that can run simultaneously.
- **Maxuuxqts**: Limits the number of instances of the BNU `uuxqt` daemon that can run simultaneously on the local system.
- **Permissions**: Specifies BNU permissions for remote systems that call or are called by the local system.
- **Poll**: Specifies when the BNU program should poll remote systems.
- **Systems**: Lists remote computers with which users of the local system can communicate using the Basic Networking Utilities (BNU) program.

**tip** File Formats

- **phones**
  - Describes connections used by the `tip` command to contact remote systems.
- **remote**
  - Describes remote systems contacted by the `tip` command.
- **.tiprc**
  - Provides initial settings of variables for the `tip` command.

TCP/IP System Management File Formats

- **3270keys**: Defines user keyboard mapping and colors for TELNET (3270).
- **Domain Cache**: Defines the root name server or servers for a DOMAIN name server host.
- **Domain Data**: Stores name resolution information for the `named` daemon.
- **Domain Local Data**: Defines the local loopback information for `named` on the name server host.
- **Domain Reverse Data**: Stores reverse name resolution information for the `named` daemon.
- **ftpusers**: Specifies local user names that cannot be used by remote FTP clients.
- **gated.conf**: Contains configuration information for the `gated` daemon.
- **gateways**: Specifies Internet routing information to the `routed` and `gated` daemons on a network.
- **hosts**: Defines the Internet Protocol (IP) name and address of the local host and specifies the names and addresses of remote hosts.
- **hosts.equiv**: Specifies remote systems that can execute commands on the local system.
- **hosts.lpd**: Specifies remote hosts that can print on the local host.
inetd.conf
Defines how the inetd daemon handles Internet service requests.

map3270
Defines keyboard mapping and colors for the tn3270 command.

.netrc
Specifies automatic login information for the ftp and rexec commands.

networks
Contains the network name file.

protocols
Defines the Internet protocols used on the local host.

rc.net
Defines host configuration for the following areas: network interfaces, host name, default gateway, and any static routes.

resolv.conf
Defines DOMAIN name server information for local resolver routines.

.rhosts
Specifies remote users that can use a local user account on a network.

services
Defines the sockets and protocols used for Internet services.

Standard Resource Record Format
telnet.conf
Defines the format of lines in the DOMAIN data files.

Translates a client's terminal-type strings into terminfo file entries.

.3270keys File Format for TCP/IP

Purpose
Defines keyboard mapping and colors for the tn and telnet command.

Description
The $HOME/.3270keys file specifies for a user a tn or telnet command key mapping that differs from the default mapping found in the /etc/3270.keys file. You can use it, for example, to make the Action key act as the Enter key.

If you are using a color display, you can also customize the colors for various 3270 display attributes by setting attributes in the .3270keys file. The default mapping in the /etc/3270.keys file is generic. The user can also load the user-defined files for specific terminal types by using the .3270keys file. The .3270keys file is specified in the user's home directory. The default background color is black. You cannot configure the background color.

The tn or telnet command first checks the $HOME directory for the .3270keys file and loads it. If the file doesn't exist, the /etc/3270.keys file is loaded. Both files, by default, end with an if statement and a list of terminal types. If the TERM environment variable matches one of the listed terminals, a second file is loaded. If the TERM variable does not match, the tn or telnet command uses the generic key bindings specified before the if statement and prints the message NOBINDINGS. This file is part of TCP/IP in Network Support Facilities.

Note: When remapping keys to customize your $HOME/.3270keys file, remember that you cannot map a 3270 function to the Esc key alone. You can specify the Esc key only in combination with another key. Also, when mapping keys, do not duplicate key sequences. For example, if you have mapped the backtab function to the ^A (the Ctrl-A key sequence), then mapping the PF1 function key to ^Aep (the Ctrl-Aep key sequence) is going to conflict with the backtab mapping.

The $HOME/.3270keys.hft File
You can also use the /usr/lpp/tcpip/samples/3270keys.hft sample file to create a $HOME/.3270keys.hft file by copying the sample file to your home directory and modifying it as necessary.
The following options can be used in the sequence field:

- \b  Backspace
- \s  Space
- \t  Tab
- \n  New line
- \r  Return
- \e  Escape
- ^  Mask next character with \037; for example, ^M.
- ~  Set high-order bit for next character.

The following are valid colors for 3270 display attributes:

- black
- blue
- red
- green
- white
- magenta
- cyan

For more information about changing the assignment of a key set, see Changing the assignment of a key set in Networks and communication management.

Note: The 3270keys.hft file supports the Attention key, which sends an IAC BREAK TELNET protocol sequence to the TELNET server on a VM or MVS system. The TELNET server is responsible for implementing the Attention key. Example 2 shows the format for binding the Attention key to the Ctrl-F12 key sequence.

Examples

1. The following example binds the Backspace key and the Tab keys:
   ```
   3270 Function   Sequence   Key
   bind   backspace   "\b"   #backspace key
   bind   tab         "\t"   #tab key
   ```
   The # (pound sign) is used to indicate comments.

2. The following example binds the Attention key to the Ctrl-F12 key sequence:
   ```
   3270 Function   Sequence   Key
   bind   attention   "\e[036q"   #attention key
   ```

Files

- /etc/3270.keys
- /etc/3270keys.hft
- /usr/lpp/tcpip/samples/3270keys.hft

Related Information

The telnet, tn, or tn3270 command.

The map3270 file format.

Changing the assignment of a key set in Networks and communication management.
acct File Format

Purpose
Provides the accounting file format for each process.

Description
The accounting files provide a means to monitor the use of the system. These files also serve as a method for billing each process for processor usage, materials, and services. The `acct` system call produces accounting files. The `/usr/include/sys/acct.h` file defines the records in these files, which are written when a process exits.

The acct structure
The `acct` structure in the `acct.h` header file contains the following fields:

- **ac_flag**: Specifies one of the following accounting flags for the process for which the accounting record is written:
  - `AFORK`: The process was created using a `fork` command but an `exec` subroutine has not yet followed. The `exec` subroutine turns off the AFORK flag.
  - `ASU`: The process used root user authority.

- **ac_stat**: Specifies the exit status. A flag that indicates how the process terminated.

- **ac_uid**: Specifies the user ID of the process for which the accounting record is written.

- **ac_gid**: Specifies the group ID of the process for which the accounting record is written.

- **ac_tty**: Specifies the terminal from which the process was started.

- **ac_wlmkey**: Holds a 64-bit numeric key representing the Workload Manager class to which the process belonged. The Workload Manager Application Programming Interface provides the `wlm_key2class` subroutine to convert the key back to a class name.

- **ac_btime**: Specifies the beginning time. The time at which the process started.

- **ac_utime**: Specifies the amount of user time, in seconds, used by the process.

- **ac_stime**: Specifies the amount of system time, in seconds, used by the process.

- **ac_etime**: Specifies the amount of time, in seconds, elapsed since the command ran.

- **ac_mem**: Specifies the average amount of memory used by the process. Every clock interrupt (or clock tick, 100 times per second), the `sys_timer` routine is called to update the user data for the current process. If the process is in user mode, both its `u_utime` value and memory usage values are incremented; otherwise, only its `u_stime` value is incremented. The `sys_timer` routine calls the `vms_rusage` routine to obtain the kilobytes of real memory being used by TEXTSEG (#1), the PRIVSEG (#2), and the big-data segments (#3-11), if used. These values are added to the total memory usage value at each clock tick during which the process is not in kernel mode. When the process ends, the `acctexit` routine computes how many clock ticks occurred while the process executed (in both user and kernel modes) and divides the total memory usage value by this number to give an average memory usage for the process. This value is recorded as a two-byte unsigned short integer.

- **ac_io**: Specifies the number of characters transferred by the process.

- **ac_rw**: Specifies the number of blocks read or written by the process.

- **ac_comm**: Specifies the name of the command that started the process. A child process created by a `fork` subroutine receives this information from the parent process. An `exec` subroutine resets this field.

The tacct Structure
The `tacct` structure, which is not part of the `acct.h` header file, represents the total accounting format used by the various accounting commands:

```
struct tacct {
    uid_t ta_uid; /* user-ID */
    char ta_name[8]; /* login name */
    float ta_cpu[2]; /* cum. CPU time, p/np (mins) */
    float ta_kcore[2]; /* cum. kcore-mins, p/np */
    float ta_io[2]; /* cum. chars xferred (512s) */
    float ta_rw[2]; /* cum. blocks read/written */
```

Chapter 2. File Formats  335
float ta_con[2]; /* cum. connect time, p/np, mins */
float ta_du; /* cum. disk usage */
long ta_qsys; /* queuing sys charges (pgs) */
float ta_fee; /* fee for special services */
long ta_pc; /* count of processes */
unsigned short ta_sc; /* count of processes */
unsigned short ta_dc; /* count of disk samples */
};

Related Information
The acctcms command, acctcom command, acctcon1 or acctcon2 command, acctdisk command,
acctmerg command, acctprc1, acctprc2, or accton command, runacct command.
The acct subroutine, fork subroutine, exec subroutine, wlm_key2class subroutine.
The Header Files Overview in AIX 5L Version 5.3 Files Reference.
The Accounting files and the System accounting in Operating system and device management.

ar File Format (Big)

Purpose
Combines several files into one. This is the default ar library archive format for the operating system.

Description
The ar (archive) file format combines several files into one. The ar command creates an archive file. The ld (link editor) command searches archive files to resolve program linkage. The /usr/include/ar.h file describes the archive file format. This file format accommodates both 32-bit and 64-bit object files within the same archive.

This is the default file format used by the ar command. To use a format portable to versions prior to AIX 4.3.0, refer to ar File Format (Small).

Fixed-Length Header
Each archive begins with a fixed-length header that contains offsets to special archive file members. The fixed-length header also contains the magic number, which identifies the archive file. The fixed-length header has the following format:

```
#define __AR_BIG__
#define AIAMAGBIG "<bigaf>n" /* Magic string */
#define SAIAMAG 8 /*Length of magic string */
struct fl_hdr /*Fixed-length header */
{
    char fl_magic[SAIAMAG]; /* Archive magic string */
    char fl_memoff[20]; /*Offset to member table */
    char fl_gstoff[20]; /*Offset to global symbol table */
    char fl_gsto64off[20]; /*Offset global symbol table for 64-bit objects */
    char fl_fstmoff[20]; /*Offset to first archive member */
    char fl_lsmoff[20]; /*Offset to last archive member */
    char fl_freeoff[20]; /*Offset to first mem on free list */
}
```

The indexed archive file format uses a double-linked list within the archive file to order the file members. Therefore, file members may not be sequentially ordered within the archive. The offsets contained in the fixed-length header locate the first and last file members of the archive. Member order is determined by the linked list.
The fixed-length header also contains the offsets to the member table, the global symbol table, and the free list. Both the member table and the global symbol table exist as members of the archive and are kept at the end of the archive file. The free list contains file members that have been deleted from the archive. When adding new file members to the archive, free list space is used before the archive file size is expanded. A zero offset in the fixed-length header indicates that the member is not present in the archive file.

**File Member Header**

Each archive file member is preceded by a file member header, which contains the following information about the file member:

```c
#define AIAFMAG "\n" /* Header trailer string*/
struct ar_hdr /* File member header*/ {
    char ar_size[20]; /* File member size - decimal */
    char ar_nxtmem[20]; /* Next member offset - decimal */
    char ar_prvmem[20]; /* Previous member offset - dec */
    char ar_date[12]; /* File member date - decimal */
    char ar_uid[12]; /* File member userid - decimal */
    char ar_gid[12]; /* File member group id - decimal */
    char ar_mode[12]; /* File member mode - octal */
    char ar_namlen[4]; /* File member name length - dec */
    union {
        char ar_name[2]; /* Start of member name */
        char ar_fmag[2]; /* AIAFMAG - string to end */
    }
    _ar_name; /* Header and member name */
};
```

The member header provides support for member names up to 255 characters long. The `ar_namlen` field contains the length of the member name. The character string containing the member name begins at the `_ar_name` field. The AIAFMAG string is cosmetic only.

Each archive member header begins on an even-byte boundary. The total length of a member header is:

```
sizeof (struct ar_hdr) + ar_namlen
```

The actual data for a file member begins at the first even-byte boundary beyond the member header and continues for the number of bytes specified by the `ar_size` field. The `ar` command inserts null bytes for padding where necessary.

All information in the fixed-length header and archive members is in printable ASCII format. Numeric information, with the exception of the `ar_mode` field, is stored as decimal numbers; the `ar_mode` field is stored in octal format. Thus, if the archive file contains only printable files, you can print the archive.

**Member Table**

A member table is always present in an indexed archive file. This table quickly locates members of the archive. The `fl_memoff` field in the fixed-length header contains the offset to the member table. The member table member has a zero-length name. The `ar` command automatically creates and updates (but does not list) the member table. A member table contains the following information:

- The number of members. This member is 20 bytes long and stored in ASCII format as a decimal number.
- The array of offsets into the archive file. The length is 20 times the number of members. Each offset is 20 bytes long and stored in ASCII format as a decimal number.
- The name string table. The size is:

  ```
  ar_size - (20 * (the number of members +1));
  ```

  that is, the size equals the total length of all members minus the length of the offsets, minus the length of the number of members.
The string table contains the same number of strings as offsets. All strings are null-terminated. Each offset from the array corresponds sequentially to a name in the string table.

Global Symbol Tables
Immediately following the member table, the archive file contains two global symbol tables. The first global symbol table locates 32-bit file members that define global symbols; the second global symbol table does the same for 64-bit file members. If the archive has no 32-bit or 64-bit file members, the respective global symbol table is omitted. The strip command can be used to delete one or both global symbol tables from the archive. The fl_gstoff field in the fixed-length header contains the offset to the 32-bit global symbol table, and the fl_gst64off contains the offset to the 64-bit global symbol table. The global symbol table members have zero-length names. The ar command automatically creates and updates, but does not list the global symbol tables. A global symbol table contains the following information:

- The number of symbols. This is 8 bytes long and can be accessed with the sgetl and sputl commands.
- The array of offsets into the archive file. The length is eight times the number of symbols. Each offset is 8 bytes long and can be accessed with the sgetl and sputl commands.
- The name-string table. The size is:
  
  \[
  \text{size} = \text{ar\_size} - (8 \times (\text{the number of symbols} + 1))
  \]

  That is, the size equals the total length of the members, minus the length of the offsets, minus the length of the number of symbols.

  The string table contains the same number of strings as offsets. All strings are null-terminated. Each offset from the array corresponds sequentially to a name in the string table.

Related Information
The a.out file format.

The ar command, ld command, strip command.

The sgetl or sputl subroutine.

ar File Format (Small)

Purpose
Describes the small indexed archive file format, in use prior to Version 4.3 of the operating system. This format is recognized by commands for backward compatability purposes only. See ar File Format (Big) for the current archive file format.

Description
The ar (archive) command combines several files into one. The ar command creates an archive file. The ld (link editor) command searches archive files to resolve program linkage. The /usr/include/ar.h file describes the archive file format. This archive format only handles 32-bit XCOFF members. The ar File Format (Big) handles both 32-bit and 64-bit XCOFF members.

Fixed-Length Header
Each archive begins with a fixed-length header that contains offsets to special archive file members. The fixed-length header also contains the magic number, which identifies the archive file. The fixed-length header has the following format:

```c
#define AIAMAG "<aiaff>\n" /* Magic string */
#define SAIAMAG 8 /* Length of magic string */
struct fl_hdr /* Fixed-length header */
{
```

338  AIX 5L Version 5.3 Files Reference
char fl_magic[SIAAMAG]; /* Archive magic string */
char fl_memoff[12];   /* Offset to member table */
char fl_gstoff[12];   /* Offset to global symbol table */
char fl_listmoff[12]; /* Offset to first archive member */
char fl_freeoff[12];  /* Offset to first mem on free list */

The indexed archive file format uses a double-linked list within the archive file to order the file members. Therefore, file members may not be sequentially ordered within the archive. The offsets contained in the fixed-length header locate the first and last file members of the archive. Member order is determined by the linked list.

The fixed-length header also contains the offsets to the member table, the global symbol table, and the free list. Both the member table and the global symbol table exist as members of the archive and are kept at the end of the archive file. The free list contains file members that have been deleted from the archive. When adding new file members to the archive, free list space is used before the archive file size is expanded. A zero offset in the fixed-length header indicates that the member is not present in the archive file.

**File Member Header**

Each archive file member is preceded by a file member header, which contains the following information about the file member:

```c
#define AIAFMAG "\n" /* Header trailer string */
struct ar_hdr /* File member header */ {
    char ar_size[12]; /* File member size - decimal */
    char ar_nxtmem[12]; /* Next member offset - decimal */
    char ar_prvmem[12]; /* Previous member offset - dec */
    char ar_date[12]; /* File member date - decimal */
    char ar_uid[12]; /* File member user id - decimal */
    char ar_gid[12]; /* File member group id - decimal */
    char ar_mode[12]; /* File member mode - octal */
    char ar_namlen[4]; /* File member name length - dec */
    union {
        char ar_name[2]; /* Start of member name */
        char ar_fmag[2]; /* AIAFMAG - string to end */
    } _ar_name; /* Header and member name */
};
```

The member header provides support for member names up to 255 characters long. The `ar_namlen` field contains the length of the member name. The character string containing the member name begins at the `_ar_name` field. The AIAFMAG string is cosmetic only.

Each archive member header begins on an even-byte boundary. The total length of a member header is:

```
sizeof (struct ar_hdr) + ar_namlen
```

The actual data for a file member begins at the first even-byte boundary beyond the member header and continues for the number of bytes specified by the `ar_size` field. The `ar` command inserts null bytes for padding where necessary.

All information in the fixed-length header and archive members is in printable ASCII format. Numeric information, with the exception of the `ar_mode` field, is stored as decimal numbers; the `ar_mode` field is stored in octal format. Thus, if the archive file contains only printable files, you can print the archive.
Member Table
A member table is always present in an indexed archive file. This table quickly locates members of the archive. The fl_memoff field in the fixed-length header contains the offset to the member table. The member table member has a zero-length name. The ar command automatically creates and updates (but does not list) the member table. A member table contains the following information:

- The number of members. This member is 12 bytes long and stored in ASCII format as a decimal number.
- The array of offsets into the archive file. The length is 12 times the number of members. Each offset is 12 bytes long and stored in ASCII format as a decimal number.
- The name string table. The size is:
  \[\text{ar\_size} - (12 \times (\text{the number of members} +1))\];
  that is, the size equals the total length of all members minus the length of the offsets, minus the length of the number of members.
  The string table contains the same number of strings as offsets. All strings are null-terminated. Each offset from the array corresponds sequentially to a name in the string table.

Global Symbol Table
If an archive file contains XCOFF object-file members that are not stripped, the archive file will contain a global symbol-table member. This global symbol table locates file members that define global symbols. The strip command deletes the global symbol table from the archive. The fl_gstoff field in the fixed-length header contains the offset to the global symbol table. The global symbol table member has a zero-length name. The ar command automatically creates and updates, but does not list the global symbol table. A global symbol table contains the following information:

- The number of symbols. This is 4 bytes long and can be accessed with the sgetl and sputl commands.
- The array of offsets into the archive file. The length is four times the number of symbols. Each offset is 4 bytes long and can be accessed with the sgetl and sputl commands.
- The name-string table. The size is:
  \[\text{ar\_size} - (4 \times (\text{the number of symbols} +1))\];
  That is, the size equals the total length of the members, minus the length of the offsets, minus the length of the number of symbols.
  The string table contains the same number of strings as offsets. All strings are null-terminated. Each offset from the array corresponds sequentially to a name in the string table.

Related Information
The a.out file format.
The ar command, ld command, strip command.
The sgetl or sputl subroutine.
The Header Files Overview in AIX 5L Version 5.3 Files Reference.

ate.def File Format
Purpose
Determines default settings for the Asynchronous Terminal Emulation (ATE) program.

Description
The ate.def file sets the defaults for use in asynchronous connections and file transfers. This file is part of Asynchronous Terminal Emulation and is created in the current directory during the first run of ATE. The ate.def file contains the default values in the ATE program uses for the following:
• Data transmission characteristics
• Local system features
• Dialing directory file
• Control keys

The first time the ATE program runs from a particular directory, it creates the `ate.def` file in that directory, with settings as follows:

```plaintext
LENGTH  8
STOP    1
PARITY  0
RATE    1200
DEVICE  tty0
INITIAL ATDT
FINAL
WAIT    0
ATTEMPTS 0
TRANSFER p
CHARACTER 0
NAME    kapture
LINEFEEDS 0
ECHO    0
VT100   0
WRITE   0
XON/XOFF 1
DIRECTORY /usr/lib/dir
CAPTURE_KEY 002
MAINMENU_KEY 026
PREVIOUS_KEY 022
```

Edit the `ate.def` file with any ASCII text editor to permanently change the values of these characteristics. Temporarily change the values of these characteristics with the ATE `alter` and `modify` subcommands, accessible from either ATE Main Menu.

**Parameters in the ate.def File**

Type parameter names in uppercase letters in the `ate.def` file. Spell the parameters exactly as they appear in the original default file. Define only one parameter per line. An incorrectly defined value for a parameter causes ATE to return a system message. However, the program continues to run using the default value.

These are the `ate.def` file parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LENGTH</td>
<td>Specifies the number of bits in a data character. This length must match the length expected by the remote system. Options: 7 or 8. Default: 8.</td>
</tr>
<tr>
<td>STOP</td>
<td>Specifies the number of stop bits appended to a character to signal that character’s end during data transmission. This number must match the number of stop bits used by the remote system. Options: 1 or 2. Default: 1.</td>
</tr>
</tbody>
</table>
PARITY
Checks whether a character is successfully transmitted to or from a remote system. Must match the parity of the remote system.

For example, if the user selects even parity, when the number of 1 bits in the character is odd, the parity bit is turned on to make an even number of 1 bits.

Options: 0 (none), 1 (odd), or 2 (even).
Default: 0.

RATE
Determines the baud rate, or the number of bits transmitted per second (bps). The speed must match the speed of the modem and that of the remote system.

Options: 50, 75, 110, 134, 150, 300, 600, 1200, 1800, 2400, 4800, 9600, or 19,200.
Default: 1200.

DEVICE
Specifies the name of the asynchronous port used to make a connection to a remote system.

Options: Locally created port names.
Default: tty0.

INITIAL
Defines the dial prefix, a string that must precede the telephone number when the user autodials with a modem. For the proper dial commands, consult the modem documentation.

Options: ATDT, ATDP, or other values, depending on the type of modem.
Default: ATDT.

FINAL
Defines the dial suffix, a string that must follow the telephone number when the user autodials with a modem. For the proper dial commands, consult the modem documentation.

Options: Blank (none) or a valid modem suffix.
Default: No default.

WAIT
Specifies the time to wait between redialing attempts. The wait period does not begin until the connection attempt times out or until it is interrupted. If the ATTEMPTS parameter is set to 0, no redial attempt occurs.

Options: 0 (none) or a positive integer designating the number of seconds to wait.
Default: 0.

ATTEMPTS
Specifies the maximum number of times the ATE program tries to redial to make a connection. If the ATTEMPTS parameter is set to 0, no redial attempt occurs.

Options: 0 (none) or a positive integer designating the number of attempts.
Default: 0.

TRANSFER
Defines the type of asynchronous protocol that transfers files during a connection.

p  pacing:
File transfer protocol controls the data transmission rate by waiting for a specified character or for a certain number of seconds between line transmissions. This helps prevent loss of data when the transmission blocks are either too large or sent too quickly for the system to process.

x  xmodem:
An 8-bit file transfer protocol to detect data transmission errors and retransmit the data.

Options: p (pacing), x (xmodem).
Default: p.
CHARACTER

Specifies the type of **pacing** protocol to be used.

*Character*

Signal to transmit a line. Select one character.

When the [send] subcommand encounters a line-feed character while transmitting data, the subcommand waits to receive the pacing character before sending the next line.

When the [receive] subcommand is ready to receive data, it sends the pacing character, then waits 30 seconds to receive data. The receive subcommand sends a pacing character again whenever it finds a carriage-return character in the data. The receive subcommand ends when it receives no data for 30 seconds.

*Interval* Number of seconds the system waits between each line it transmits. The value of the Interval variable must be an integer. The default value is 0, indicating a pacing delay of 0 seconds.

Default: 0.

**NAME**

File name for incoming data (capture file).

Options: A valid file name less than 40 characters long.

Default: The *kapture* file.

**LINEFEEDS**

Adds a line-feed character after every carriage-return character in the incoming data stream.

Options: 1 (on) or 0 (off).

Default: 0.

**ECHO**

Displays the user’s typed input.

For a remote computer that supports echoing, each character sent returns and displays on the screen. When the ECHO parameter is on, each character is displayed twice: first when it is entered, and again when it returns over a connection. When the ECHO parameter is off, each character displays only when it returns over the connection.

Options: 1 (on) or 0 (off).

Default: 0.

**VT100**

The local console emulates a DEC VT100 terminal so DEC VT100 codes can be used with the remote system. With the VT100 parameter off, the local console functions like a workstation.

Options: 1 (on) or 0 (off).

Default: 0.

**WRITE**

Captures incoming data and routes it to the file specified in the NAME parameter as well as to the display. Carriage-return or line-feed combinations are converted to line-feed characters before they are written to the capture file. In an existing file, data is appended to the end of the file.

Note: The CAPTURE_KEY (usually the Ctrl-B key sequence) can be used to toggle capture mode on or off during a connection.

Options: 1 (on) or 0 (off).

Default: 0.
XON/XOFF Controls data transmission at a port as follows:
- When an Xoff signal is received, transmission stops.
- When an Xon signal is received, transmission resumes.
- An Xoff signal is sent when the receive buffer is nearly full.
- An Xon signal is sent when the buffer is no longer full.

Options: 1 (On) or 0 (Off).

Default: 1.

DIRECTORY Names the file that contains the user’s dialing directory.

Default: the /usr/lib/dir file.

CAPTURE_KEY Defines the control key sequence that toggles capture mode. When pressed, the CAPTURE_KEY (usually the Ctrl-B key sequence) starts or stops capturing (saving) the data that is displayed on the screen during an active connection.

Options: Any ASCII control character.

Default: ASCII octal 002 (STX).

MAINMENU_KEY Defines the control key sequence that returns the ATE Connected Main Menu so the user can issue a command during an active connection. The MAINMENU_KEY (usually the Ctrl-V key sequence) functions only from the connected state.

Options: Any ASCII control character.

Default: ASCII octal 026 (SYN).

PREVIOUS_KEY Defines the control key sequence that displays the previous screen anytime during the program. The screen displayed varies, depending on the screen in use when the user presses PREVIOUS_KEY (usually the Ctrl-R key sequence).

Options: Any ASCII control character.

Default: ASCII octal 022 (DC2). The ASCII control character is mapped to the interrupt signal.

Notes:
1. Changing or remapping may be necessary if control keys conflict across applications. For example, if the control keys mapped for the ATE program conflict with those in a text editor, remap the ATE control keys.
2. The ASCII control character selected may be in octal, decimal, or hexadecimal format, as follows:

   - **octal**: 000 through 037. The leading zero is required.
   - **decimal**: 0 through 31.
   - **hexadecimal**: 0x00 through 0x1F. The leading 0x is required. The x may be uppercase or lowercase.

Examples
To change characteristics of ATE emulation, create an **ate.def** file that defines those characteristics.

For example, to change the RATE to 300 bps, the DEVICE to tty3, the TRANSFER mode to x (*xmodem* protocol), and the DIRECTORY to my.dir, create the following **ate.def** file in the directory running the ATE program:

```
RATE 300
DEVICE tty3
TRANSFER x
DIRECTORY my.dir
```

The time the ATE program starts from that directory, the program uses the defined values.
Related Information
The ate command.

The alter subcommand, connect subcommand, directory subcommand, modify subcommand, receive subcommand, send subcommand.

Asynchronous communications, Asynchronous Terminal Emulation, ate.def configuration file, Setting up an ATE dialing directory in Networks and communication management.

audit File Format

Purpose
Describes the auditing data structures.

Description
The /usr/include/sys/audit.h file contains structure and constant definitions for the auditing system commands, subroutines, and daemons:

Audit Bin Format
The format of the audit bin is described by the aud_bin structure. An audit trail consists of a sequence of bins, each of which must start with a bin head and end with a bin tail. The aud_bin structure contains the following fields:

bin_magic: The magic number for the bin (0xf0f0).
bin_version: The version number for the bin (0).
bin_tail: Indicates whether the bin describes the audit trail head or tail:
0: Identifies the bin header.
1: Identifies the bin end (tail).
2: Identifies the trail end.
bin_len: The (unpacked) length of the bin's records. A nonzero value indicates that the bin has a tail record.
bin_plen: The current length of the bin’s record (might be packed).
bin_time: The time at which the head or tail was written.
bin_reserved1: Not currently used.
bin_reserved2: Not currently used.

Audit Class Format
The format of the audit class is described by the audit_class structure, which contains the following fields:

ae_name: A pointer to the name of the audit class.
ae_list: A pointer to a list of null-terminated audit event names for this audit class. The list is ended by a null name (a leading null byte or two consecutive null bytes).
Note: Event and class names are limited to 15 significant characters.
ae_len: The length of the event list in the ae_list member. This length includes the terminating null bytes. On an AUDIT_SET operation, the caller must set this member to indicate the actual length of the list (in bytes) pointed to by ae_list. On an AUDIT_GET or AUDIT_LOCK operation, the auditevents subroutine sets this member to indicate the actual size of the list.
Audit Object Format
The format of the audit object is described by the `o_event` structure, which contains the following fields:

- **`o_type`**: Specifies the type of the object, in terms of naming space. Currently, only one object-naming space is supported:
  - `AUDIT_FILE`: Denotes the file system naming space.

- **`o_name`**: Specifies the name of the object.

- **`o_event`**: Specifies any array of event names to be generated when the object is accessed. Note that event names are currently limited to 16 bytes, including the trailing null. The index of an event name in this array corresponds to an access mode. Valid indexes are defined in the `audit.h` file and include the following:
  - `AUDIT_READ`
  - `AUDIT_WRITE`
  - `AUDIT_EXEC`

*Note*: The C++ compiler will generate a warning indicating that `o_event` is defined both as a structure and a field within that structure. Although the `o_event` field can be used within C++, the warning can by bypassed by defining `O_EVENT_RENAME`. This will replace the `o_event` field with `o_event_array`. `o_event` is the default field.

Audit Record Format
Each audit record consists of a list of fixed-length event identifiers, each of which can be followed by a variable-length tail. The format of the audit record is described by the `aud_rec` structure, which contains the following fields to identify the event:

- **`ah_magic`**: Magic number for audit record.
- **`ah_length`**: The length of the tail portion of the audit record.
- **`ah_event[16]`**: The name of the event and a null terminator.
- **`ah_result`**: An indication of whether the event describes a successful operation. The values for this field are:
  - `0`: Indicates successful completion.
  - `1`: Indicates a failure.
  - `>1`: An `errno` value describing the failure.

The `aud_rec` structure also contains the following fields to identify the user and the process:

- **`ah_ruid`**: The real user ID; that is, the ID number of the user who created the process that wrote this record.
- **`ah_luid`**: The login ID of the user who created the process that wrote this record.
- **`ah_name[16]`**: The program name of the process, along with a null terminator.
- **`ah_pid`**: The process ID of the process that wrote this record.
- **`ah_ppid`**: The process ID of the parent of this process.
- **`ah_time`**: The time in seconds at which this audit record was written.
- **`ah_ntime`**: The nanoseconds offset from `ah_time`.

The record tail follows this header information.

Related Information
The `audit`, `auditcat`, `auditpr`, `auditselect`, `auditstream` command.

The `auditbin` daemon.
The **audit** subroutine, **auditbin** subroutine, **auditevents** subroutine, **auditlog** subroutine, **auditobj** subroutine, **auditproc** subroutine, **auditwrite** subroutine.

Header Files Overview in *AIX 5L Version 5.3 Files Reference*.

---

**bootptab File Format**

**Purpose**
Default configuration database for the Internet Boot Protocol server (**bootpd**).

**Description**
The **bootpd** configuration file contains entries for clients that use the **bootpd** daemon to get boot information. This file may be modified using the System Management Interface Tool (SMIT) to configure a Diskless client or the file may be modified manually.

The client host information consists of case-sensitive tag symbols used to represent host parameters. These host parameter declarations are separated by : (colon). For example:

```
HostName: Tg=Value:Tg=Value:Tg=Value
```

where:

**HostName**

Specifies the name of a BOOTP client. This must always be the first field in the entry.

The **bootpd** daemon attempts to send the entire host name as it is specified in this field. However, if the host name does not fit into the reply packet, the name is shortened to the host field (up to the first period, if present) and tried again. An arbitrarily truncated host name is never sent. If nothing reasonable fits, nothing is sent.

**Guidelines and Restrictions**

- Blank lines and lines beginning with # are ignored when the file is read.
- Host entries are separated from one another by new lines; a single host entry may be extended over multiple lines if the lines end with a backslash (\). However, individual host entries must not exceed 1024 characters.
- Lines in the configuration file may be longer than 80 characters.
- Tags can be displayed in any order, with the following exceptions:
  - The host name must be the first field in an entry, and
  - The hardware type must precede the hardware address.

**Related Information**
The **bootpd** Daemon.
The CHARMAP Section

The CHARMAP section of the charmap file maps symbolic character names to code points. All supported code sets have the portable character set as a proper subset. Only symbols that are not defined in the portable character set must be defined in the CHARMAP section. The portable character set consists of the following character symbols (listed by their standardized symbolic names) and encodings:

<table>
<thead>
<tr>
<th>Symbol Name</th>
<th>Code (hexadecimal)</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;NUL&gt;</td>
<td>000</td>
</tr>
<tr>
<td>&lt;SOH&gt;</td>
<td>001</td>
</tr>
<tr>
<td>&lt;STX&gt;</td>
<td>002</td>
</tr>
<tr>
<td>&lt;ETX&gt;</td>
<td>003</td>
</tr>
<tr>
<td>&lt;EOT&gt;</td>
<td>004</td>
</tr>
<tr>
<td>&lt;ENQ&gt;</td>
<td>005</td>
</tr>
<tr>
<td>&lt;ACK&gt;</td>
<td>006</td>
</tr>
<tr>
<td>&lt;alert&gt;</td>
<td>007</td>
</tr>
<tr>
<td>&lt;backspace&gt;</td>
<td>008</td>
</tr>
<tr>
<td>&lt;tab&gt;</td>
<td>009</td>
</tr>
<tr>
<td>&lt;new-line&gt;</td>
<td>00A</td>
</tr>
<tr>
<td>&lt;vertical-tab&gt;</td>
<td>00B</td>
</tr>
<tr>
<td>&lt;form-feed&gt;</td>
<td>00C</td>
</tr>
<tr>
<td>&lt;carriage-return&gt;</td>
<td>00D</td>
</tr>
<tr>
<td>&lt;SO&gt;</td>
<td>00E</td>
</tr>
<tr>
<td>&lt;SI&gt;</td>
<td>00F</td>
</tr>
<tr>
<td>&lt;DLE&gt;</td>
<td>010</td>
</tr>
<tr>
<td>&lt;DC1&gt;</td>
<td>011</td>
</tr>
<tr>
<td>&lt;DC2&gt;</td>
<td>012</td>
</tr>
<tr>
<td>&lt;DC3&gt;</td>
<td>013</td>
</tr>
<tr>
<td>&lt;DC4&gt;</td>
<td>014</td>
</tr>
<tr>
<td>&lt;NAK&gt;</td>
<td>015</td>
</tr>
<tr>
<td>&lt;SYN&gt;</td>
<td>016</td>
</tr>
<tr>
<td>&lt;ETB&gt;</td>
<td>017</td>
</tr>
<tr>
<td>&lt;CAN&gt;</td>
<td>018</td>
</tr>
<tr>
<td>&lt;EM&gt;</td>
<td>019</td>
</tr>
<tr>
<td>&lt;SUB&gt;</td>
<td>01A</td>
</tr>
<tr>
<td>&lt;ESC&gt;</td>
<td>01B</td>
</tr>
<tr>
<td>&lt;IS4&gt;</td>
<td>01C</td>
</tr>
<tr>
<td>&lt;IS3&gt;</td>
<td>01D</td>
</tr>
<tr>
<td>&lt;IS2&gt;</td>
<td>01E</td>
</tr>
<tr>
<td>&lt;IS1&gt;</td>
<td>01F</td>
</tr>
<tr>
<td>&lt;space&gt;</td>
<td>020</td>
</tr>
<tr>
<td>&lt;exclamation-mark&gt;</td>
<td>021</td>
</tr>
<tr>
<td>&lt;quotation-mark&gt;</td>
<td>022</td>
</tr>
<tr>
<td>&lt;number-sign&gt;</td>
<td>023</td>
</tr>
<tr>
<td>&lt;dollar-sign&gt;</td>
<td>024</td>
</tr>
<tr>
<td>&lt;percent&gt;</td>
<td>025</td>
</tr>
<tr>
<td>&lt;ampersand&gt;</td>
<td>026</td>
</tr>
<tr>
<td>&lt;apostrophe&gt;</td>
<td>027</td>
</tr>
<tr>
<td>&lt;left-parenthesis&gt;</td>
<td>028</td>
</tr>
<tr>
<td>&lt;right-parenthesis&gt;</td>
<td>029</td>
</tr>
<tr>
<td>Symbol Name</td>
<td>Code (hexadecimal)</td>
</tr>
<tr>
<td>----------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>&lt;asterisk&gt;</td>
<td>02A</td>
</tr>
<tr>
<td>&lt;plus-sign&gt;</td>
<td>02B</td>
</tr>
<tr>
<td>&lt;comma&gt;</td>
<td>02C</td>
</tr>
<tr>
<td>&lt;hyphen&gt;</td>
<td>02D</td>
</tr>
<tr>
<td>&lt;period&gt;</td>
<td>02E</td>
</tr>
<tr>
<td>&lt;slash&gt;</td>
<td>02F</td>
</tr>
<tr>
<td>&lt;zero&gt;</td>
<td>030</td>
</tr>
<tr>
<td>&lt;one&gt;</td>
<td>031</td>
</tr>
<tr>
<td>&lt;two&gt;</td>
<td>032</td>
</tr>
<tr>
<td>&lt;three&gt;</td>
<td>033</td>
</tr>
<tr>
<td>&lt;four&gt;</td>
<td>034</td>
</tr>
<tr>
<td>&lt;five&gt;</td>
<td>035</td>
</tr>
<tr>
<td>&lt;six&gt;</td>
<td>036</td>
</tr>
<tr>
<td>&lt;seven&gt;</td>
<td>037</td>
</tr>
<tr>
<td>&lt;eight&gt;</td>
<td>038</td>
</tr>
<tr>
<td>&lt;nine&gt;</td>
<td>039</td>
</tr>
<tr>
<td>&lt;colon&gt;</td>
<td>03A</td>
</tr>
<tr>
<td>&lt;semi-colon&gt;</td>
<td>03B</td>
</tr>
<tr>
<td>&lt;less-than&gt;</td>
<td>03C</td>
</tr>
<tr>
<td>&lt;equal-sign&gt;</td>
<td>03D</td>
</tr>
<tr>
<td>&lt;greater-than&gt;</td>
<td>03E</td>
</tr>
<tr>
<td>&lt;question-mark&gt;</td>
<td>03F</td>
</tr>
<tr>
<td>&lt;commercial-at&gt;</td>
<td>040</td>
</tr>
<tr>
<td>&lt;A&gt;</td>
<td>041</td>
</tr>
<tr>
<td>&lt;B&gt;</td>
<td>042</td>
</tr>
<tr>
<td>&lt;C&gt;</td>
<td>043</td>
</tr>
<tr>
<td>&lt;D&gt;</td>
<td>044</td>
</tr>
<tr>
<td>&lt;E&gt;</td>
<td>045</td>
</tr>
<tr>
<td>&lt;F&gt;</td>
<td>046</td>
</tr>
<tr>
<td>&lt;G&gt;</td>
<td>047</td>
</tr>
<tr>
<td>&lt;H&gt;</td>
<td>048</td>
</tr>
<tr>
<td>&lt;I&gt;</td>
<td>049</td>
</tr>
<tr>
<td>&lt;J&gt;</td>
<td>04A</td>
</tr>
<tr>
<td>&lt;K&gt;</td>
<td>04B</td>
</tr>
<tr>
<td>&lt;L&gt;</td>
<td>04C</td>
</tr>
<tr>
<td>&lt;M&gt;</td>
<td>04D</td>
</tr>
<tr>
<td>&lt;N&gt;</td>
<td>04E</td>
</tr>
<tr>
<td>&lt;O&gt;</td>
<td>04F</td>
</tr>
<tr>
<td>&lt;P&gt;</td>
<td>050</td>
</tr>
<tr>
<td>&lt;Q&gt;</td>
<td>051</td>
</tr>
<tr>
<td>&lt;R&gt;</td>
<td>052</td>
</tr>
<tr>
<td>&lt;S&gt;</td>
<td>053</td>
</tr>
<tr>
<td>&lt;T&gt;</td>
<td>054</td>
</tr>
<tr>
<td>&lt;U&gt;</td>
<td>055</td>
</tr>
<tr>
<td>&lt;V&gt;</td>
<td>056</td>
</tr>
<tr>
<td>&lt;W&gt;</td>
<td>057</td>
</tr>
<tr>
<td>&lt;X&gt;</td>
<td>058</td>
</tr>
<tr>
<td>&lt;Y&gt;</td>
<td>059</td>
</tr>
<tr>
<td>&lt;Z&gt;</td>
<td>05A</td>
</tr>
<tr>
<td>&lt;left-bracket&gt;</td>
<td>05B</td>
</tr>
<tr>
<td>&lt;backsplash&gt;</td>
<td>05C</td>
</tr>
<tr>
<td>&lt;right-bracket&gt;</td>
<td>05D</td>
</tr>
<tr>
<td>&lt;circumflex&gt;</td>
<td>05E</td>
</tr>
<tr>
<td>&lt;underscore&gt;</td>
<td>05F</td>
</tr>
</tbody>
</table>
### Symbol Name  
**<grave-accent>**  
**<a>**  
**<b>**  
**<c>**  
**<d>**  
**<e>**  
**<f>**  
**<g>**  
**<h>**  
**<i>**  
**<j>**  
**<k>**  
**<l>**  
**<m>**  
**<n>**  
**<o>**  
**<p>**  
**<q>**  
**<r>**  
**<s>**  
**<t>**  
**<u>**  
**<v>**  
**<w>**  
**<x>**  
**<y>**  
**<z>**  
**<left-brace>**  
**<vertical-line>**  
**<right-brace>**  
**<tilde>**  
**<DEL>**

### Code (hexadecimal)  
060  
061  
062  
063  
064  
065  
066  
067  
068  
069  
06A  
06B  
06C  
06D  
06E  
06F  
070  
071  
072  
073  
074  
075  
076  
077  
078  
079  
07A  
07B  
07C  
07D  
07E  
07F

The **CHARMAP** section contains the following sections:

- The **CHARMAP** section header.
- An optional special symbolic name-declarations section. The symbolic name and value must be separated by one or more blank characters. The following are the special symbolic names and their meanings:

  - `<code_set_name>`
    - Specifies the name of the coded character set for which the `charmap` file is defined. This value determines the value returned by the `nl_langinfo` subroutine. The `<code_set_name>` must be specified using any character from the portable character set, except for control and space characters.

  - `<mb_cur_max>`
    - Specifies the maximum number of bytes in a multibyte character for the encoded character set. Valid values are 1 to 4. The default value is 1.

  - `<mb_cur_min>`
    - Specifies the minimum number of bytes in a multibyte character for the encoded character set. Since all supported code sets have the portable character set as a proper subset, this value must be 1.

  - `<escape_char>`
    - Specifies the escape character that indicates encodings in hexadecimal or octal notation. The default value is a `\` (backslash).
<comment_char> Specifies the character used to indicate a comment within a charmap file. The default value is a # (pound sign). With the exception of optional comments following a character symbol encoding, comments must start with a comment character in the first column of a line.

- Character set mapping statements for the defined code set.
  Each statement in this section defines a symbolic name for a character encoding. A character symbol begins with the < (less-than) character and ends with the > (greater-than) character. The characters between the < (less-than) and > (greater-than) can be any characters from the portable character set, except for control and space characters. The > (greater-than) character may be used if it is escaped with the escape character (as specified by the <escape_char> special symbolic name). A character symbol cannot exceed 32 characters in length.
  The format of a character symbol definition is:
  
  \texttt{<char_symbol> encoding}
  \texttt{optional comment}

  An encoding is specified as one or more character constants, with the maximum number of character constants specified by the \texttt{<mb_cur_max>} special symbolic name. The \texttt{localedef} command supports decimal, octal, and hexadecimal constants with the following formats:

  \begin{itemize}
    \item \texttt{hexadecimal constant \xddd}
    \item \texttt{octal constant \oddd}
    \item \texttt{decimal constant \ddd}
  \end{itemize}

  Some examples of character symbol definitions are:

  \begin{itemize}
    \item \texttt{<A> \d65 \text{ decimal constant}}
    \item \texttt{<B> \x42 \text{ hexadecimal constant}}
    \item \texttt{<j0101> \x81 \d254 \text{ mixed hex and decimal constants}}
  \end{itemize}

  A range of one or more symbolic names and corresponding encoding values may also be defined, where the nonnumeric prefix for each symbolic name is common, and the numeric portion of the second symbolic name is equal to or greater than the numeric portion of the first symbolic name. In this format, a symbolic name value consists of zero or more nonnumeric characters followed by an integer of one or more decimal digits. This format defines a series of symbolic names. For example, the string \texttt{<j0101>...<j0104>} is interpreted as the \texttt{<j0101>}, \texttt{<j0102>}, \texttt{<j0103>}, and \texttt{<j0104>} symbolic names, in that order.

  In statements defining ranges of symbolic names, the encoded value is the value for the first symbolic name in the range. Subsequent symbolic names have encoding values in increasing order. For example:

  \texttt{<j0101>...<j0104> \d129\d254}

  This character set mapping statement is interpreted as follows:

  \begin{itemize}
    \item \texttt{<j0101> \d129\d254}
    \item \texttt{<j0102> \d129\d255}
    \item \texttt{<j0103> \d130\d0}
    \item \texttt{<j0104> \d130\d1}
  \end{itemize}

  Symbolic names must be unique, but two or more symbolic names can have the same value.

- The END CHARMAP section trailer.

**Examples**

The following is an example of a portion of a possible CHARMAP section from a charmap file:

\begin{verbatim}
CHARMAP
<code_set_name> ISO8859-1
<mb_cur_max> 1
<mb_cur_min> 1
\end{verbatim}
The CHARSETID Section

The CHARSETID section maps the code points within the code set to a character set ID. The CHARSETID section contains the following sections:

- The CHARSETID section header.
- Character set ID mappings for the defined code sets.
- The END CHARSETID section trailer.

Character set ID mappings are defined by listing symbolic names or code points for symbolic names and their associated character set IDs. The following are possible formats for a character set ID mapping statement:

- `<character_symbol>`
- `<character_symbol>...<character_symbol>`
- `character_constant`
- `character_constant...character_constant`

The `<character_symbol>` used must have previously been defined in the CHARMAP section. The `character_constant` must follow the format described for the CHARMAP section.

Individual character set mappings are accomplished by indicating either the symbolic name (defined in the CHARMAP section or the portable character set) followed by the character set ID, or the code point associated with a symbolic name followed by the character set ID value. Symbolic names and code points must be separated from a character set ID value by one or more blank characters. Ranges of code points can be mapped to a character set ID value by indicating appropriate combinations of symbolic names and code point values as endpoints to the range, separated by ... (ellipsis) to indicate the intermediate characters, and followed by the character set ID for the range. The first endpoint value must be less than or equal to the second end point value.

Examples

The following is an example of a portion of a possible CHARSETID section from a charmap file:

```
CHARSETID
<space>...<nobreakspace> 0
<tilde>...<y-diaeresis> 1
END CHARSETID
```

Related Information

- Locale Definition Source File Format
- Locale Method Source File Format

For specific information about the locale categories and keywords, see the LC_COLLATE category, LC_CTYPE category, LC_MESSAGES category, LC_MONETARY category, LC_NUMERIC category, and LC_TIME category.
The `locale` command, `localedef` command.

For information on converting data between code sets, see `Converters Overview for System Management`, `National Language Support Overview for System Management`, `Understanding the Character Set Description (charmap) Source File` in `AIX 5L Version 5.3 National Language Support Guide and Reference`.

---

### Core File Format

#### Purpose
Contains an image of a process at the time of an error.

#### Description
A core file is created in the current directory when various errors occur. Errors such as memory-address violations, illegal instructions, bus errors, and user-generated quit signals, commonly cause this core dump. The core file that is created contains a memory image of the terminated process. If the faulty process is multi-threaded and the current core size ulimit is less than what is required to dump the data section, then only the faulting thread stack area is dumped from the data section.

**Note:** Beginning with AIX 5.1, the core dump file can be given a unique name in the format. In AIX 5.2, this can only be enabled by an environment variable. For AIX 5.3, see the `chcore` command:

```bash
core.pid.ddhhmmss
```

where:
- `pid`: process id
- `dd`: Day of the Month
- `hh`: Hours
- `mm`: Minutes
- `ss`: Seconds

Default behavior is the same as in previous versions of AIX.

Any shared memory allocated by the process may also be optionally omitted from the core file. This data is only omitted if the `CORE_NOSHM` environment variable is exported. The default is to include all shared memory in the core file. The `CORE_NOSHM` variable can be set to any value.

Unique core file naming will only be enabled if the environment variable `CORE_NAMING` is exported. The default name is `core` as in previous releases. The value of the variable should be set to `true`.

A process with a saved user ID that differs from the real user ID does not produce a memory image. The same holds true for the group ID (GID) and effective group ID. The contents of a core dump are organized sequentially in the core file as follows:

- **Core header**
  - Defines basic information about the core dump, and contains offsets that locate the remainder of the core dump information.
- **ldinfo structures**
  - Defines loader information.
- **mstsave structures**
  - Defines kernel thread state information. Since the faulting thread `mstsave` structure is directly saved in the core header, additional structures are saved here only for multi-threaded programs.
- **Default user stack**
  - Contains a copy of the user stack at the time of the core dump.
- **Default data area**
  - (Optional) Contains the user data section.
- **Memory mapped regions**
  - (Optional) Contains the anonymously mapped regions.
- **vm_info structures**
  - (Optional) Contains offset and size information for memory mapped regions.
By default, the user data is, anonymously mapped regions, and `vm_info` structures are not included in a core dump. This partial core dump includes the current process stack, thread stack, the thread `mstsave` structures, the user structure, and the state of the registers at the time of the fault. A partial core dump contains sufficient information for a stack traceback. The size of a core dump can also be limited by the `setrlimit` subroutine.

To enable a full core dump, set the `SA_FULLDUMP` flag in the `sigaction` subroutine for the signal that is to generate a full core dump. If this flag is set when the core is dumped, the data section is, anonymously mapped regions, and `vm_info` structures are included in the core dump.

The format of the core header is defined by the `core_dump` structure (in the `core.h` header file), which is organized as follows:

<table>
<thead>
<tr>
<th>Field Type</th>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>char</td>
<td>c_signo</td>
<td>The number of the signal which caused the error</td>
</tr>
<tr>
<td>char</td>
<td>c_flag</td>
<td>A bit field which describes the core dump type. The meanings of the bits are as follows:</td>
</tr>
<tr>
<td></td>
<td>FULL_CORE</td>
<td>core contains the data sections (0x01)</td>
</tr>
<tr>
<td></td>
<td>CORE_VERSION_1</td>
<td>core was generated by AIX Version 4 or higher (0x02)</td>
</tr>
<tr>
<td></td>
<td>MSTS_VALID</td>
<td>core contains mstsave structures (0x04)</td>
</tr>
<tr>
<td></td>
<td>CORE_BIGDATA</td>
<td>core contains big data (0x08)</td>
</tr>
<tr>
<td></td>
<td>UBLOCK_VALID</td>
<td>core contains the u_block structure (0x10)</td>
</tr>
<tr>
<td></td>
<td>USTACK_VALID</td>
<td>core contains the user stack (0x20)</td>
</tr>
<tr>
<td></td>
<td>LE_VALID</td>
<td>core contains at least one module (0x40)</td>
</tr>
<tr>
<td></td>
<td>CORE_TRUNC</td>
<td>core was truncated (0x80)</td>
</tr>
<tr>
<td>ushort</td>
<td>c_entries</td>
<td>The number of core dump modules</td>
</tr>
<tr>
<td>struct ld_info *</td>
<td>c_tab</td>
<td>The offset to the beginning of the core table</td>
</tr>
<tr>
<td>caddr_t</td>
<td>c_stack</td>
<td>The offset to the beginning of the user stack</td>
</tr>
<tr>
<td>int</td>
<td>c_size</td>
<td>The size of the user stack</td>
</tr>
<tr>
<td>struct mstsave</td>
<td>c_mst</td>
<td>A copy of the faulting mst</td>
</tr>
<tr>
<td>struct user</td>
<td>c_u</td>
<td>A copy of the user structure</td>
</tr>
<tr>
<td>int</td>
<td>c_rmsts</td>
<td>The number of <code>mstsave</code> structures referenced by the <code>c_msts</code> field</td>
</tr>
<tr>
<td>struct mstsave *</td>
<td>c_msts</td>
<td>The offset to the other threads' <code>mstsave</code> structures</td>
</tr>
<tr>
<td>int</td>
<td>c_datasize</td>
<td>The size of the data region</td>
</tr>
<tr>
<td>caddr_t</td>
<td>c_data</td>
<td>The offset to user data</td>
</tr>
<tr>
<td>int</td>
<td>c_vmregions</td>
<td>The number of anonymously mapped regions</td>
</tr>
<tr>
<td>struct vm_info *</td>
<td>c_vmm</td>
<td>The offset to the start of the <code>vm_info</code> table</td>
</tr>
</tbody>
</table>
Related Information

The `param.h` file.

The `adb` command, `dbx` command, and `chcore` command.

The `raise` subroutine, `setrlimit` subroutine, `setuid` subroutine, `sigaction` subroutine.

---

**core File Format (AIX 4.2)**

**Purpose**

Contains an image of a 32-bit process at the time of an error.

**Description**

A core file is created in the current directory when various errors occur. Errors such as memory-address violations, illegal instructions, bus errors, and user-generated quit signals commonly cause this core dump. The core file that is created contains a memory image of the terminated process. A process with a saved user ID that differs from the real user ID does not produce a memory image. The contents of a core dump are organized sequentially in the core file as follows:

- **Core header**: Defines basic information about the core dump, and contains offsets which locate the remainder of the core dump information.
- **ldinfo structures**: Defines loader information.
- **mstsave structures**: Defines kernel thread state information. Since the faulting thread mstsave structure is directly saved in the core header, additional structures are saved here only for multi-threaded programs.
- **Default user stack**: Contains a copy of the user stack at the time of the core dump.
- **Default data area**: (Optional) Contains the user data section.
- **Memory mapped regions**: (Optional) Contains the anonymously mapped regions.
- **vm_info structures**: (Optional) Contains offset and size information for memory mapped regions.

The `core_dump` structure, defined by the `core.h` file, occurs at the beginning of a core file. The `core_dump` structure includes the following fields:

<table>
<thead>
<tr>
<th>Field Type</th>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>char</td>
<td>c_signo</td>
<td>The number of the signal which caused the error</td>
</tr>
<tr>
<td>Field Type</td>
<td>Field Name</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>char</td>
<td>c_flag</td>
<td>A bit field which describes the core dump type. The meanings of the bits are as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>FULL_CORE</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>core contains the data sections (0x01)</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>CORE_VERSION_1</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>core was generated by AIX Version 4 or higher (0x02)</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>MSTS_VALID</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>core contains mstsave structures (0x04)</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>CORE_BIGDATA</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>core contains big data (0x08)</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>UBLOCK_VALID</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>core contains the u_block structure (0x10)</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>USTACK_VALID</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>core contains the user stack (0x20)</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>LE_VALID</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>core contains at least one module (0x40)</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>CORE_TRUNC</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>core was truncated (0x80)</td>
</tr>
</tbody>
</table>

| ushort     | c_entries    | The number of core dump modules |
| struct id_info * | c_tab    | The offset to the beginning of the core table |
| caddr_t    | c_stack     | The offset to the beginning of the user stack |
| int        | c_size      | The size of the user stack |
| struct mstsave | c_mst     | A copy of the faulting mst |
| struct user | c_u        | A copy of the user structure |
| int        | c_nmsts     | The number of mstsave structures referenced by the c_msts field |
| struct mstsave * | c_msts    | The offset to the other threads’ mstsave structures |
| int        | c_datasize  | The size of the data region |
| caddr_t    | c_data      | The offset to user data |
| int        | c_vmregions | The number of anonymously mapped regions |
| struct vm_info * | c_vmm | The offset to the start of the vm_info table |

The c_u field contains the **user structure** (a copy of the actual u_block), which includes the registers as they existed at the time of the fault.

The ld_info structure and then the user-mode stack follow the u_block in the core dump.

By default, the user data, anonymously mapped regions, and vm_info structures are not included in a core dump. This partial core dump includes the current thread stack, the thread mstsave structures, the user structures, and the state of the registers at the time of the fault. A partial core dump contains sufficient information for a stack traceback. The size of a core dump can also be limited by the setrlimit subroutine.

To enable a full core dump, set the **SA_FULLDUMP** flag in the sigaction subroutine for the signal that is to generate a full core dump. If this flag is set when the core is dumped, the data section, anonymously mapped regions, and vm_info structures are included in the core dump.
Related Information

The `param.h` file.

The `adb` command, `dbx` command.

The `raise` subroutine, `setrlimit` subroutine, `setuid` subroutine, `sigaction` subroutine.

The Header Files Overview in AIX 5L Version 5.3 Files Reference defines header files, describes how they are used, and lists several header files for which information is provided in this documentation.

---

core File Format (AIX 4.3)

Purpose

Contains an image of a 32-bit or 64-bit process at the time of an error.

Description

A core file is created in the current directory when various errors occur. Errors such as memory-address violations, illegal instructions, bus errors, and user-generated quit signals commonly cause this core dump. The core file that is created contains a memory image of the terminated process. A process with a saved user ID that differs from the real user ID does not produce a memory image. The contents of a core dump are organized sequentially in the core file as follows:

- **Core header**: Defines basic information about the core dump, and contains offsets that locate the remainder of the core dump information.
- **ldinfo structures**: Defines loader information.
- **thrdctx structures**: Defines kernel thread state information. Since the faulting thread thrdctx structure is directly saved in the core header, additional structures are saved here only for multi-threaded programs.
- **segregion structures**: Contains the address, size, and type for shared memory segments of the faulting process.
- **Default user stack**: Contains a copy of the user stack at the time of the core dump.
- **Default data area**: (Optional) Contains the user data section.
- **vm_infox structures**: (Optional) Contains offset and size information for memory mapped regions.
- **Memory mapped regions**: (Optional) Contains the memory mapped regions.

The core_dumpx structure, defined by the `core.h` file, occurs at the beginning of a core file. The core_dumpx structure includes the following fields:

<table>
<thead>
<tr>
<th>Field Type</th>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>char</td>
<td>c_signo</td>
<td>The number of the signal that caused the error</td>
</tr>
</tbody>
</table>
### Field Type | Field Name | Description
---|---|---
char | c_flag | A bit field that describes the core dump type. The meanings of the bits are as follows:
  - **FULL_CORE**: core contains the data sections (0x01)
  - **CORE_VERSION_1**: core was generated by AIX Version 4 or higher (0x02)
  - **MSTS_VALID**: core contains mstsave structures (0x04)
  - **CORE_BIGDATA**: core contains big data (0x08)
  - **UBLOCK_VALID**: core contains the u_block structure (0x10)
  - **USTACK_VALID**: core contains the user stack (0x20)
  - **LE_VALID**: core contains at least one module (0x40)
  - **CORE_TRUNC**: core was truncated (0x80)
ushort | c_entries | The number of core dump modules
int | c_version | Core file format version
unsigned long long | c_fdsinfox | Offset to fd region in file
unsigned long long | c_loader | Offset to loader region in file
unsigned long long | c_lsize | Size of the loader region
unsigned long long | c_n_thr | Number of elements in thread table
unsigned long long | c_thr | Offset to thread context table
unsigned long long | c_segs | Number of elements in segregion
unsigned long long | c_segregnion | Offset to start of segregion table
unsigned long long | c_stack | Offset of user stack in file
unsigned long long | c_stackorg | Base address of user stack region
unsigned long long | c_size | Size of user stack region
unsigned long long | c_data | Offset to user data region
unsigned long long | c_dataorg | Base address of user data region
unsigned long long | c_datasize | Size of user data region
unsigned long long | c_sdorg | Base address of privately loaded region
unsigned long long | c_sdszize | Size of user privately loaded region
unsigned long long | c_vmregions | Number of anonymously mapped areas
unsigned long long | c_vmm | Offset of start of vm_infox table
struct thrdctx | c_fltt | Faulting thread’s context and thread data
struct userx | c_u | Representation of the user struct and process data

The `c_flag` field in the core dump contains the `thrdctx` structure of the faulting thread. The `thrdctx` structure includes the thread data and registers as they existed at the time of the fault. The format of the thread context structure is defined by the `thrdctx` structure (in the `core.h` header file) as follows:
The \texttt{c\_u} field follows this information in the core dump. The \texttt{c\_u} field contains the \texttt{userx} structure including the user structure fields, and the process data as they existed at the time of the fault. The format of the process information structure is defined by the \texttt{userx} structure (in the \texttt{core.h} header file) as follows:

\begin{verbatim}
userx procsinfo64 process data (in procinfo.h header file)
\end{verbatim}

The \texttt{id\_info} structure and then the \texttt{thrdctx} structures of the other threads (if the process is multi-threaded) follow in the core dump.

The \texttt{segregion} structure and then the user-mode stack follow in the core dump.

The \texttt{segregion} structure contains the information about a shared memory region of the faulting process.

\begin{verbatim}
segregion addr segment start address
size size of the segment
segflags type of the document
\end{verbatim}

The first three fields of the \texttt{core\_dumpx} header in AIX 4.3 are the same as that of the \texttt{core\_dump} header in AIX 4.2. However, the \texttt{c\_entries} are always zero on AIX 4.3 systems to distinguish them from the AIX 4.2 core file formats. Further, the \texttt{pi\_flags2} field of the \texttt{procsinfo64} structure determines if the core file is of a 32-bit process or a 64-bit process.

The AIX 4.3 operating system can be forced to create core files in a AIX 4.2 core file format via the SMIT tool. However, this enforcement is valid only for 32-bit processes.

By default, the user data, anonymously mapped regions, and \texttt{vm\_infox} structures are not included in a core dump. This partial core dump includes the current thread stack, the thread \texttt{thrdctx} structures, the user structure, and the state of the registers at the time of the fault. A partial core dump contains sufficient information for a stack traceback. The size of a core dump can also be limited by the \texttt{setrlimit} or \texttt{setrlimit64} subroutine.

To enable a full core dump, set the \texttt{SA\_FULLDUMP} flag in the \texttt{sigaction} subroutine for the signal that is to generate a full core dump. If this flag is set when the core is dumped, the user data section, \texttt{vm\_infox}, and anonymously mapped region structures are included in the core dump.

Related Information

The \texttt{param.h} file.

The \texttt{adb} command, \texttt{dbx} command.

The \texttt{raise} subroutine, \texttt{setrlimit} and \texttt{setrlimit64} subroutines, \texttt{setuid} subroutine, \texttt{sigaction} subroutine.

The \texttt{Header Files Overview} in AIX 5L Version 5.3 Files Reference defines header files, describes how they are used, and lists several header files for which information is provided in this documentation.
**cpio File Format**

**Purpose**
Describes the copy in/out (**cpio**) archive file.

**Description**
The **cpio** utility backs up and recovers files. The files are saved on the backup medium in the **cpio** format.

When the **cpio** command is used with the **-c** flag, the header for the **cpio** structure reads as follows:

```c
sscanf(Chdr,"%6ho%6ho%6ho%6ho%6ho%6ho%6ho%6ho%6ho%6ho%11lo%6ho%11lo%s",
&Hdr.h_magic, &Hdr.h_dev, &Hdr.h_ino, &Hdr.h_mode,
&Hdr.h_uid, &Hdr.h_gid, &Hdr.h_nlink, &Hdr.h_rdev,
&Longtime, &Hdr.h_namesize, &Longfile, &Hdr.h_name);
```

Longtime and Longfile are equivalent to Hdr.h_mtime and Hdr.h_filesize, respectively. The contents of each file, and other items describing the file, are recorded in an element of the array of structures with varying lengths.

**Note:** Files saved with the **-c** flag must be restored with the **-c** flag.

When the **-c** flag of the **cpio** command is not used, the header structure contains the following fields:

- **h_magic**: Contains the constant octal 070707 (or 0x71c7).
- **h_dev**: Device that contains a directory entry for this file.
- **h_ino**: I-node number that identifies the input file to the file system.
- **h_mode**: Mode of the input file, as defined in the **mode.h** file.
- **h_uid**: User ID of the owner of the input file.
- **h_gid**: Group ID of the owner of the input file.
- **h_nlink**: Number of links that are connected to the input file.
- **h_rdev**: ID of the remote device from which the input file is taken.
- **h_mtime**: Time when data was last modified. For remote files, this field contains the time at the server. This time can be changed by the **creat**, **fclearf**, **truncate**, **mknod**, **openx**, **pipe**, **utime**, or **writex** subroutine.
- **h_namesize**: Length of the path name, including the terminating null byte.
- **h_filesize**: Length of the file in bytes. This is the length of the data section that follows the header structure.
- **h_name**: Null-terminated path name. The length of the path name, including the null byte, is indicated by the **n** variable, where **n** equals \((h\_namesize \div 2) + h\_namesize\). That is, the **n** variable is equal to the **h_namesize** field if the **h_namesize** field is even. If the **h_namesize** field is odd, the **n** variable is equal to the **h_namesize** field + 1.

The last record of the archive always contains the name **TRAILER!!!**. Special files, directories, and the trailer are recorded with the **h_filesize** field equal to 0.

**Related Information**
The **mode.h** file, **stat.h** file.

The **cpio** command, **find** command.
The `fclear` subroutine, `truncate` or `ftruncate` subroutine, `mknod` subroutine, `open`, `openx`, or `creat` subroutine, `pipe` subroutine, `scanf`, `fscanf`, `sscanf`, `wscanf` subroutine, `utime` subroutine, `write`, `writex`, `writev`, or `writevx` subroutine.

The **Header Files Overview** in *AIX 5L Version 5.3 Files Reference* defines header files, describes how they are used, and lists several of the header files for which information is provided in this documentation.

# cronlog.conf File

## Purpose
Specifies the default *cron* configuration file for logging information.

## Description
The *cronlog.conf* configuration file informs the *cron* daemon where and how to log information. If you do not use the `-f` flag, the *cron* daemon reads the default `/etc/cronlog.conf` configuration file. If *cron* fails to open the configuration file, it continues with `/var/adm/cron/log`. The *cron* daemon ignores blank lines and lines beginning with a `#` (pound sign).

## Format

- **logfile** Specifies the full path name of the log file. If the file does not exist, it is created with permission 664. Do not create the log file in the “/” file system. If the *cron* daemon cannot create the name of the log file, the daemon creates a log of its activities in the `/var/adm/cron/log` file, and all of the features below it are turned off.

- **size** This keyword limits the size of a *logfile*. It is followed by a number and either a `k` (kilobytes) or `m` (megabytes). The default and minimum size is 30K. If the size value is invalid then rotation feature is turned off.

- **rotate** This keyword specifies the total number of rotated files. It is followed by a number. If a number is not specified then there are unlimited number of rotated files. If the *rotate* keyword is not present, then no rotation is done. If specified, minimum of number of rotated files is 2.

- **compress** This keyword specifies that the saved rotated files will be compressed. If keyword is not present then no compression is done.

- **archive** This keyword specifies that the saved rotated files will be copied to a directory. It is followed by the directory name. If archive is not specified and rotation is valid, files are rotated in the directory of *logfile*.

- **quiet** This keyword specifies that *cron* logging will be disabled.

## Examples

1. To log information in the directory `/home/user`, log file *cron.out*, size of 2M, total number of rotated files = 4, archive in `/usr/home`, compress archive files, create the configuration file as follows:

   ```
   logfile=/home/user/cron.out
   size=2m
   rotate=4
   archive=/usr/home
   compress
   ```

2. To log information in directory `/home/user`, log file *cron.out*, size of 50k, unlimited log files, archive in `/usr/home`, compress archive files, create the configuration file as follows:
3. To log information in directory `/home/user`, log file `cron.out`, size of 50k, total number of rotated files = 4, create the configuration file as follows:
   
   ```
   logfile=/home/user/cron.out
   size=50k
   rotate
   archive=/usr/home
   compress
   ```

4. To log information in directory `/home/user`, log file `cron.out`, size of 50k, total number of rotated files = 4, archive in `/usr/home`, archive files without compression, no logging of cron jobs, create the configuration file as follows:

   ```
   logfile=/home/user/cron.out
   size=50K
   rotate=4
   archive=/usr/home
   quiet
   ```

   In this case, `cron.out` will contain only debug or error messages.

**Related Information**

The [cron](#) daemon.

**Devices File Format for BNU**

**Purpose**

Contains information about devices on the local system that can establish a connection to a remote computer using the Basic Networking Utilities (BNU) program.

**Description**

The `/etc/uucp/Devices` file and its augmentations and alternatives specified in the `/etc/uucp/ Sysfiles` file contains information about the devices on the local system that can establish a connection to a remote computer using the Basic Networking Utilities (BNU) program. This file includes information for hardwired, telephone, and TCP/IP communication links.

**Note:** Only someone with root user authority can edit the `Devices` file, which is owned by the `uucp` login ID.

**Fields in the Devices File**

The `Devices` file must contain a description of each device on the local system that can establish a remote connection using the BNU program. Each line in the `Devices` file includes the following fields:

- **Type**
  - Typically specifies the type of hardwired or automatic calling unit (ACU) device.
- **Line**
  - Specifies the device name for the port.
- **Line2**
  - Specifies the dialer name if the `Line` entry specifies an 801 dialer.
- **Class**
  - Typically specifies the transmission speed.
- **Dialer-Token Pairs**
  - Specifies a particular type of autodialer (modem) and the token (a defined string of characters) that is passed to the dialer. Valid entries for this field are defined in the `/etc/uucp/Dialers` file.

The fields appear on the line as follows:

`Type Line Line2 Class Dialer-Token Pairs`
Every field of a line in the **Devices** file must contain an entry. If a field does not apply to the particular type of device or system, use a - (minus sign) as a placeholder.

Lines in the **Devices** file cannot wrap. Each entry must be on only one line in the file. However, the **Devices** file can contain blank lines and comment lines. Comment lines begin with a # (pound sign). Blank lines are ignored.

**Type Field:** Enter one of the following keywords in this field:

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACU</td>
<td>Use this keyword, entered in uppercase letters, if your site connects multiple systems over the telephone network with automatic calling units (autodialers or modems).</td>
</tr>
<tr>
<td>Direct</td>
<td>Use this keyword, beginning with an uppercase D, if your site uses hardwired lines to connect multiple systems.</td>
</tr>
<tr>
<td>TCP</td>
<td>Use this keyword, in uppercase letters, if your site uses TCP/IP.</td>
</tr>
<tr>
<td>SystemName</td>
<td>Enter the name of a particular remote system hardwired to the local system. The <em>SystemName</em> keyword is the name assigned to each individual system, such as hera, zeus, or merlin.</td>
</tr>
</tbody>
</table>

This field corresponds to the **Type** field in the */etc/uucp/Systems* file.

**Line Field:** The device name for the line, or port, used in the communication link is inserted here. For example, use the appropriate device name for a hardwired line, such as tty1. For a line connected to an ACU (a modem), use a device name appropriate to the dialer, such as tty1 or tty2. For a TCP connection, enter a minus sign as a placeholder.

**Line2 Field:** Unless you are using an 801 dialer, use a - (minus sign) in this field as a placeholder. If you are using an 801 dialer, put the device name of the 801 ACU in this field. For example, if the entry in the **Type** field is ACU and the **Line** field entry (specifying the modem) is tty1, the **Line2** field entry (specifying the 801 dialer for the modem) might be tty3 or tty4.

**Note:** The **Line2** field is used only to support older modems that require 801-type dialers. The modem is plugged into one serial port, and the 801 dialer is plugged into a separate serial port.

**Class Field:** For an ACU or a hardwired line, the **Class** field can be the speed of the device. In this case, for a hardwired line, use the transmission rate of the device connecting the two systems. For a telephone connection, use the speed at which the ACU transmits data, such as 300 or 1200 bps.

This field can also contain a letter with a speed (for example, C1200 or 1200) to differentiate between classes of dialers. For example, some offices have more than one telephone network, one for internal use and one for external communications. In such a case, it is necessary to distinguish which lines should be used for each connection.

The **Class** field in the **Devices** file is matched against the **Class** field in the */etc/uucp/Systems* file. For example, if the **Systems** file entry for system hera is:

```
hera Any ACU 1200 3-3-5-2 oigin: nuucp ssword: oldoaktree
```

BNU searches for an entry in the **Devices** file with a **Type** of ACU and a **Class** of 1200.

Some devices can be used at several specific speeds. In this case, make multiple entries for the device, specifying each speed on a separate line in the **Devices** file. If BNU cannot connect at the first speed, it will try the successive speeds.

If a device can be used at any speed, type the word Any in the **Class** field. Note that the A in Any must be uppercase.

For a TCP/IP connection, enter a - (minus sign) as a placeholder.
**Dialer-Token Pair Field:** The Dialer-Token Pair field specifies a particular type of autodialer (modem) and the token (a defined string of characters) that is passed to the dialer. Valid entries for this field are defined in the `/etc/uucp/Dialers` file.

For a hardwired connection, enter the word direct (note the lowercase d) as the Dialer entry and leave the Token entry blank.

For a telephone connection, enter the type of dialer and the token that is passed to that modem. The Token field entry is either a telephone number or a predefined string used to reach the dialer.

For a telephone connection, enter one of the following as the Dialer field entry:

<table>
<thead>
<tr>
<th>Entry</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>hayes</td>
<td>A Hayes dialer.</td>
</tr>
<tr>
<td>Other Dialers</td>
<td>Other dialers that you can specify by including the relevant information in the <code>/etc/uucp/Dialers</code> file.</td>
</tr>
<tr>
<td>TCP</td>
<td>A TCP/IP connection. Enter TCP in the Dialer field entry if you have also entered TCP in the Type field.</td>
</tr>
</tbody>
</table>

Each Dialer field entry included as part of a Dialer-Token Pair field in the Devices file has a corresponding entry in the Dialers file.

If the Token field entry represents a telephone number, enter one of the following in the Token field to specify how the BNU program should use the telephone number listed in the `/etc/uucp/Systems` file:

<table>
<thead>
<tr>
<th>Entry</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>\D</td>
<td>The default token in a Dialer-Token Pair field. The \D token specifies that the BNU program should take the phone number listed in the <code>/etc/uucp/Systems</code> file and pass it to the appropriate dialer script (entry) in the <code>/etc/uucp/Dialers</code> file, without including a dial-code abbreviation.</td>
</tr>
<tr>
<td>\T</td>
<td>This token instructs the BNU program to process the phone number by including the data specified in the <code>/etc/uucp/Dialcodes</code> file. Note: If you are using dial-code abbreviations specified in the Dialcodes file for certain telephone numbers, you must enter the \T string as the token in those entries in the Dialers file.</td>
</tr>
<tr>
<td>blank</td>
<td>Leaving the Token field blank is the same as entering \D, so a blank is usually sufficient as a token if you have included complete telephone numbers in the <code>/etc/uucp/Systems</code> file.</td>
</tr>
</tbody>
</table>

If the Token field does not represent a telephone number, enter the predefined string necessary to reach the dialer.

### Examples

**Setting Up Entries for Hardwired Connections**

To set up a Device file entry specifying a port and a remote system, make an entry as follows:

```
Direct tty1 - 1200 direct
zeus tty1 - 1200 direct
```

The Type field lists Direct (for a direct connection) in the first part and zeus (the name of the remote system) in the second part. The local system is connected to system zeus by way of device tty1, which is listed in the Line field in both parts of the example.

The Line2 field contains actual data only when the entry specifies a certain type of telephone connection. A - (minus sign) is used as a placeholder in other types of connections, as in this example. This device transmits at a rate of 1200 bps, which is listed in the Class field in both parts of the example. The word direct in the Dialer field portion of the Dialer-Token Pair field indicates that this is a direct connection.
Setting Up Entries for Autodialer Connections

1. For a standard Hayes modem that can be used at only one baud rate, make an entry as follows:
   ACU tty2 - 1200 hayes

   The Type field is specified as ACU. The Line field is specified with the device name tty2. Because this modem is not an 801 dialer, a - (minus sign) is used as a placeholder in the Line2 field. The Class field entry is a transmission rate of 1200 baud. The Dialer field part of the Dialer-Token Pair field is specified as a hayes modem, and the Token field part is left blank.

2. To specify a standard Hayes modem that can be used at different baud rates, make an entry as follows:
   ACU tty3 - 1200 hayes
   ACU tty3 - 300 hayes

   These two lines specify the same modem, a hayes, which can be used at either 1200 or 300 baud, as specified in the Class field. The modem is connected to a device named tty3 (the Line field), and the Line2 field contains the - (minus sign) placeholder. The Dialer field part of the Dialer-Token Pair field is specified as a hayes modem, and the Token field is left blank.

3. To specify a standard Hayes modem that can be used at any baud rate, make an entry as follows:
   ACU tty2 - Any hayes

   These two lines specify a hayes modem that can be used at any baud rate, as specified by the word Any entered in the Class field. Note that the word Any must be entered with an uppercase A.

4. To specify a connection using a standard 801 dialer, make an entry as follows:
   ACU tty4 tty5 1200 801
   ACU tty6 tty7 300 801

   In these entries, the ACU entries are connected to devices named tty4 and tty6, specified in the Line field. In both cases, there is an entry in the Line2 field because a standard 801 autodialer is specified in the Dialer-Token Pair field. Because 801 is specified as the dialer in these two examples, the Line2 field must contain the device names of the 801 ACUs. The Class field entry specifies a transmission rate of 1200 baud for the first example and 300 for the second. The Token field part of the Dialer-Token Pair field is blank.

Setting Up the Entry for Use with TCP/IP

If your site is using the TCP/IP system, enter the following in the Devices file:
TCP - - - TCP

TCP is specified in the Type field. minus signs are used as placeholders in the Line, Line2, and Class fields. TCP is specified as the Dialer field entry, with the Token entry left blank.

Setting Up Entries for Both Local and Remote Systems

The following examples illustrate the entries needed in the Devices file for both local and remote systems in order for the two systems to communicate using the BNU program.

1. To configure a hardwired connection, note the following information.
   The following entries configure local and remote Devices files for a hardwired connection between systems zeus and hera, where zeus is considered the local system and hera the remote system. The hardwired device on system zeus is tty1; on system hera, it is tty2.

   The Devices file on system zeus contains the following entry in order to connect to the remote system, hera:
   Direct tty1 - 1200 direct
   hera tty1 - 1200 direct

   The Devices file on system hera contains the following entry for communications with system zeus:
   Direct tty2 - 1200 direct
   zeus tty2 - 1200 direct
2. To configure a telephone connection, note the following information.

These files are set up to connect systems venus and merlin over a telephone line using modems. System venus is considered the local system, and system merlin is considered the remote system.

On both systems, the device tty1 is hooked to a Hayes modem at 1200 baud. Both computers include partial phone numbers in their /etc/uucp/Systems files and dialing codes in their /etc/uucp/Dialcodes files.

The Devices file on system venus contains the following entry for the connection to system merlin:

ACU tty1 - 1200 hayes \T

The Devices file on system merlin contains the following entry for the connection to system venus:

ACU tty1 - 1200 hayes \T

Files

/etc/uucp/ directory Contains all the configuration files for BNU, including the Devices file.
/etc/uucp/Dialcodes file Contains dialing code abbreviations.
/etc/uucp/Devices file Specifies initial handshaking on a connection.
/etc/uucp/Systems file Specifies possible alternative or augmentative files for /etc/uucp/Devices.
/etc/uucp/Sysfiles file

Related Information

The cu command, uucp command, uucpadm command, uuto command, uux command.

The uucico daemon, uuxqt daemon.

Editing Devices File for Hardwired Connections, Editing Devices File for Autodialer Connection, Editing Devices File for TCP/IP, Configuring BNU in Networks and communication management.

Dialcodes File Format for BNU

Purpose

Contains the initial digits of telephone numbers used to establish remote connections over a phone line.

Description

The /etc/uucp/Dialcodes file contains the initial digits of telephone numbers used by the Basic Networking Utilities (BNU) program to establish remote connections over a phone line. The Dialcodes file simplifies entries in the /etc/uucp/Systems file for sites where a number of device phone numbers have the same prefix.

If users at your site communicate regularly by way of telephone lines and modems to multiple systems all located at the same remote site, or to multiple systems located at different remote sites, use the dial-code abbreviations in the /etc/uucp/Systems file rather than entering the complete phone number of each remote modem in that file.

The Dialcodes file contains dial-code abbreviations and partial phone numbers that complete the telephone entries in the /etc/uucp/Systems file. Entries in the Dialcodes file contain an alphabetic prefix attached to a partial phone number that may include the following information in the order listed:

- Codes for an outside line
- Long-distance access codes
- A 1 (one) plus the area code (if the modem is out of the local area)
- The three-digit exchange number
The relevant alphabetic prefix (representing the partial phone number), together with the remaining four digits of that number, is then entered in the *Phone* field in the */etc/uucp/Systems* file.

Following is the form of an entry in a *Dialcodes* file:

*DialCodeAbbreviation DialingSequence*

The *DialCodeAbbreviation* part of the entry is an alphabetic prefix containing up to 8 letters, established when setting up the dialing-code listing. The *DialingSequence* is composed of all the digits in the number that precede the actual four-digit phone number.

**Notes:**

1. If your site uses only a relatively small number of telephone connections to remote systems, include the complete phone numbers of the remote modems in the */etc/uucp/Systems* file rather than use dial-code abbreviations.
2. Enter each prefix *only once* in the *Dialcodes* file. When you have set up a dial-code abbreviation, use that prefix in all relevant entries in the */etc/uucp/Systems* file.
3. Only someone with root user authority can edit the *Dialcodes* file, which is owned by the *uucp* program login ID.

**Example**

The *Dialcodes* file on system *venus* contains the following dial-code prefix for use with a number in the */etc/uucp/Systems* file:

```
local 9=445
```

The *Systems* file on system *venus* contains the following entry for system *zeus*, including a phone number and a dialing prefix:

```
zeus Any ACU 1200 local8784 in:--in: uzeus word: thunder
```

When BNU on system *venus* dials system *zeus*, BNU uses the expanded telephone number 9=4458784.

**Files**

<table>
<thead>
<tr>
<th>Directory/File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>/etc/uucp</em> directory</td>
<td>Contains all the configuration files for BNU, including the <em>Dialcodes</em> file.</td>
</tr>
<tr>
<td><em>/etc/uucp/Devices</em> file</td>
<td>Contains information about available devices.</td>
</tr>
<tr>
<td><em>/etc/uucp/Dialers</em> file</td>
<td>Specifies initial handshaking on a connection.</td>
</tr>
<tr>
<td><em>/etc/uucp/Systems</em> file</td>
<td>Describes accessible remote systems. Specifies possible files used instead of <em>/etc/uucp/System</em> file, <em>/etc/uucp/Devices</em> file, and <em>/etc/uucp/Dialers</em> file.</td>
</tr>
</tbody>
</table>

**Related Information**


[Configuring BNU](#) and [Understanding the BNU File and Directory Structure](#) in *Networks and communication management*.

**Dialers File Format for BNU**

**Purpose**

Lists modems used for Basic Networking Utilities (BNU) remote communications links.
Description

The /etc/uucp/Dialers file and its surrogates, specified in the /etc/uucp/Sysfiles file, lists the modems (dialers) used by the Basic Networking Utilities (BNU) program and specifies the initial handshaking necessary to establish remote communications links. Handshaking is a series of expect-send sequences that specify the initial communications that occur on a link before it is ready to send or receive data. Using the handshaking, the local and remote systems confirm that they are compatible and configured to transfer data.

The Dialers file(s) contains entries for each autodialer that is included in the /etc/uucp/Devices file or one of its surrogate files. Surrogate file are specified in the /etc/uucp/Sysfiles file. It also contains entries specifying no handshaking for direct hardware links (the direct entry) and TCP/IP links (the TCP entry). The first field of the Dialers file, which specifies the dialer, is matched to the fifth field of the Devices file, the Dialer-Token Pair field, to determine handshaking when making a connection.

Note: Only someone with root user authority can edit the Dialers file, which is owned by the uucp login ID.

Fields in a Dialers File

Every modem (dialer) is listed on a line by itself in the Dialers file. Each line consists of three groups of information: the Dialer Name field, the Dial Tone and Wait Characters field, and the Handshaking field.

Dialer Name Field: The first field in a Dialers file, the Dialer Name field, specifies the type of autodialer (modem) used in the connection. It matches the fifth field, the Dialer-Token Pair field, in the Devices file(s). When a particular device is used to make a connection, BNU uses the Dialer-Token Pair field in the Devices file(s) to find the handshaking entry in the Dialers file(s).

If your system has direct hardware connections to one or more remote systems, include an entry with a Dialer Name of direct. Similarly, if your system uses TCP/IP to connect to one or more other systems, include an entry with a Dialer Name of TCP. These entries correspond, respectively, to the word direct and the word TCP in the Dialer-Token Pairs field of entries in a Devices file. Omit the Dial Tone and Wait Characters field and the Handshaking field, since no handshaking is needed on these connections.

Dial Tone and Wait Characters Field: The second field, the Dial Tone and Wait Characters field, consists of two sets of two characters, for a total of four entries. These characters comprise a translation string. In the actual phone number of the remote modem, the first character in each string is mapped to the second character in that set.

<table>
<thead>
<tr>
<th>Entry</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>=,-,-,</td>
<td>Translate the telephone number. Any = (equal sign) represents wait for dial tone and any - (minus sign) represents pause.</td>
</tr>
<tr>
<td>&quot;&quot;</td>
<td>Wait for nothing; continue with the rest of the string.</td>
</tr>
<tr>
<td>WAIT=n</td>
<td>Enter this before any send string in the Dialers file, where n is the number of seconds to wait before timing out.</td>
</tr>
</tbody>
</table>

This field generally translates the = and - characters into whatever the dialer uses for wait for dial tone and pause.

For direct and TCP entries, omit this field.

Handshaking Field: The handshaking, or dialer negotiations, consists of an expect-send sequence of ASCII strings. This sequence is given in the Handshaking field, which comprises the remainder of the entry. This string is generally used to pass telephone numbers to a modem, or to make a connection to another system on the same data switch as the local system. The string tells the cu or ct program or the
The handshaking characters include the following key sequences:

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>\c</td>
<td>Suppress new line (\n)</td>
</tr>
<tr>
<td>\D</td>
<td>Raw phone number</td>
</tr>
<tr>
<td>\T</td>
<td>Translated phone number</td>
</tr>
<tr>
<td>\N</td>
<td>Null character (\0)</td>
</tr>
<tr>
<td>\b</td>
<td>Backspace</td>
</tr>
<tr>
<td>\n</td>
<td>New line</td>
</tr>
<tr>
<td>\r</td>
<td>Carriage return</td>
</tr>
<tr>
<td>\s</td>
<td>Space</td>
</tr>
<tr>
<td>\t</td>
<td>Tab</td>
</tr>
<tr>
<td>\</td>
<td>Backslash</td>
</tr>
<tr>
<td>\E</td>
<td>Turn echo check on</td>
</tr>
<tr>
<td>\e</td>
<td>Turn echo check off</td>
</tr>
<tr>
<td>\d</td>
<td>Delay two seconds</td>
</tr>
<tr>
<td>\p</td>
<td>Pause about 1/4 second</td>
</tr>
<tr>
<td>\K</td>
<td>Generate a break on the line</td>
</tr>
<tr>
<td>\M</td>
<td>Set tty setting CLOCAL on</td>
</tr>
<tr>
<td>\m</td>
<td>Turn tty setting CLOCAL off</td>
</tr>
</tbody>
</table>

For direct and TCP entries, omit this field.

Examples

Setting Up Entries in a Dialers File

1. The following example lists several entries in a typical Dialers file:

   ```
   hayes =",", " \dAT\r\c OK \pATDT\t
   \r\c CONNECT
   penril =W-P "" \d > s\p9\c }-\w\p\r\ds\p9\c-)
   y/c : \E\t
   P > 9\c OK
   ventel =8-\c "" \r\p \r\p-\r\p-9 $ <K\D\p\r\r>\c ;ONLINE!
   vadic =K-K "" \005\p >-\005\p-9 D\p BER? \E\n
   LINE
direct
TCP
   ```

   **Note:** In a Dialers file, each entry must be entirely on one line.

   Notice that the next-to-last entry in the preceding example consists only of the word direct. This entry indicates that hardwired connections do not require any handshaking. Similarly, the last entry, TCP, indicates that TCP/IP connections require no handshaking.

2. The following example interprets the first line in the preceding Dialers file. This is a standard entry that may be included in your Dialers file with modifications for use at your site.

   ```
   hayes =",", " \dAT\r\c OK \pATDT\t
   \r\c CONNECT
   ```

   The first two sequences (",","") comprise the Dial Tone and Wait Characters field. The remaining strings comprise the Handshaking field. Following is an explanation of how each entry affects the action of the dialer.
<table>
<thead>
<tr>
<th>Entry</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;,,&quot;</td>
<td>Wait for nothing; continue with the rest of the string.</td>
</tr>
<tr>
<td>\dAT</td>
<td>Delay; then send AT (the Hayes Attention prefix).</td>
</tr>
<tr>
<td>\r\c</td>
<td>Send a carriage return (r) followed by a new line (c).</td>
</tr>
<tr>
<td>OK</td>
<td>Wait for OK from the remote modem, signaling that the first part of the string has executed.</td>
</tr>
<tr>
<td>\pATDT</td>
<td>Pause (p); then send ATDT. AT is the Hayes Attention prefix, D represents a dialing signal, and T represents a touch-tone dial tone.</td>
</tr>
<tr>
<td>\T</td>
<td>Send the telephone number, which is specified in the Systems file, with dial-code translation from the Dialcodes file.</td>
</tr>
<tr>
<td>\r\c</td>
<td>Send a carriage return and a new line following the number.</td>
</tr>
<tr>
<td>CONNECT</td>
<td>Wait for CONNECT from the remote modem, signaling that the modems are connected at the baud rate specified in the Devices file.</td>
</tr>
</tbody>
</table>

**Note:** If you need to modify this example for use at your site and are unsure about the appropriate entries in the handshaking string, refer to the documentation that accompanied the modems you are including in the Dialers file.

**Setting Up the Direct Entry**

If your BNU configuration includes hardwired connections, a Dialers file must contain a direct entry, as follows:

```
=direct
```

This entry indicates that hardwired connections do not require any handshaking. It corresponds to the word direct in the Dialer-Token Pairs field of entries for hardwired devices in a Devices file (see the /etc/uucp/Devices file).

**Setting Up the TCP/IP Entry**

If your BNU configuration includes TCP/IP connections, the Dialers file used by the uucico service must contain a TCP entry, as follows:

```
TCP
```

This entry indicates that TCP/IP connections do not require any handshaking. It corresponds to the word TCP in the Dialer-Token Pairs field of entries for TCP/IP connections in the uucico service Devices file(s).

**Setting Up Entries for Both Local and Remote Systems**

The following example illustrates the entries needed in the Dialers file to correspond to entries in the Devices file for both local and remote systems so that the two systems can communicate using the BNU program.

These files are set up to connect systems venus and merlin over a telephone line using modems. System venus is considered the local system, and system merlin is considered the remote system. On both systems, the device tty1 is hooked to a hayes modem at 1200 baud.

- The Devices file on system venus contains the following entry for the connection to remote system merlin:
  ```
  ACU tty1 - 1200 hayes
  ```
- The Dialers file on system venus contains the following entry for its modem:
  ```
  hayes =,,-, \dAT\r\c OK \pATDT\r\c
  ```
- The Devices file on system merlin contains the following entry for the connection to system venus:
  ```
  ACU tty1 - 1200 hayes
  ```
- The Dialers file on system merlin contains the following entry for its modem:
Note: The Dialers file and Devices file for the system venus and merlin can be files other than \etc/uucp/Dialers and \etc/uucp/Devices. Use of the \etc/uucp/Sysfiles file enables a system administrator to allow the use of one or more files on each system to replace or augment the \etc/uucp/Dialers and \etc/uucp/Devices file. See the [Sysfiles Files Format for BNU] in AIX 5L Version 5.3 Files Reference.

Troubleshooting Connection Problems

Note: The Dialer and Systems files discussed in the section can be files other than \etc/uucp/Dialers and \etc/uucp/Systems. See the [Sysfiles Files Format for BNU] in AIX 5L Version 5.3 Files Reference.

When establishing a connection between a local and a remote system using a telephone line and modem, the BNU program consults the Dialers file. (The BNU program also checks the Systems file to make sure it contains a listing for the specified remote computer.) If users report a faulty connection, use the uucico command to debug the connection problem. For example, if users are experiencing difficulties connecting to remote system venus, issue the following command:

/usr/sbin/uucp/uucico -r1 -svenus -x9

where -r1 specifies the server mode, -svenus the name of the remote system to which you are trying to connect, and -x9 the debug level that produces the most detailed debugging information.

Expect-send debugging output produced by the uucico command can come either from information in the Dialers file or from information in the Systems file. If the relevant line in the Dialers file is not set up correctly for the specified modem, the BNU program will probably display the following error message:

DIALER SCRIPT FAILED

If the dialer script fails, verify the following:

• Make sure that both the local and the remote modems are turned on, that they are both set up correctly, and that the telephone number of the remote modem is correct.
• Check the Dialers file and make sure the information is correctly specified for the local modem. If possible, also check the Dialers file on the remote system.
• Check the documentation that came with your modem to make sure you have used the correct expect-send sequence characters in the Dialers file.

Files

\etc/uucp directory
\etc/uucp/Devices file
\etc/uucp/Dialcodes file
\etc/uucp/Systems file
\etc/uucp/Sysfiles file

Contains all the configuration files for BNU, including the Dialers file.
Contains information about available devices.
Contains dialing code abbreviations.
Describes accessible remote systems.
Specifies possible alternative files for \etc/uucp/System, \etc/uucp/Dialers, and \etc/uucp/Devices.

Related Information

The \ct command, \cu command, uukick command, \uutry command, \Uutry command.

The uucico daemon.

Configuring BNU, Monitoring a BNU remote connection, Debugging BNU login failures using the uucico daemon, [BNU File and Directory Structure] in Networks and communication management.
Dialing Directory File Format for ATE

Purpose
Lists phone numbers used to establish modem connections.

Description
The ATE dialing directory file lists phone numbers that the Asynchronous Terminal Emulation (ATE) uses to establish remote connections by modem.

Users name the dialing directory file with any valid file name and place it in any directory where read and write access is owned. Edit the dialing directory file with any ASCII text editor. The default dialing directory file is the /usr/lib/dir file.

The connect and directory subcommands of ATE access the dialing directory file. Use the connect command to use numbers that are not in the dialing directory file. Use the directory subcommand to view the dialing directory.

Users can have more than one dialing directory. To change the dialing directory file the ATE program uses, modify the ate.def file in the current directory.

Note: The dialing directory file can contain up to 20 lines (one entry per line). ATE ignores subsequent lines.

Format of Dialing Directory File Entries
The dialing directory file is similar to a page in a telephone book. This file contains entries for the remote systems called with the ATE program. The format of a dialing directory entry is:

Name Phone Rate Length StopBit Parity Echo Linefeed

The fields must be separated by at least one space. More spaces can be used to make each entry easier to read. The fields are:

Name
Identifies a telephone number. The name can be any combination of 20 or fewer characters. Use the _ (underscore) instead of a blank between words in a name, for example, data_bank.

Phone
The telephone number to be dialed. The number can be up to 40 characters. Consult the modem documentation for a list of acceptable digits and characters. For example, if a 9 must be dialed to access an outside line, include a 9 and a , (comma) before the telephone number as follows: 9,1112222.

Note: Although the telephone number can be up to 40 characters long, the directory subcommand displays only the first 26 characters.

Rate
Transmission or baud rate in bits per second (bps). Determines the number of characters transmitted per second. Select a baud rate that is compatible with the communication line being used. The following are acceptable rates: 50, 75, 110, 134, 150, 300, 600, 1200, 1800, 2400, 4800, 9600, or 19,200.

Length
Number of bits that make up a character. The entry for the Length field can be 7 or 8.

StopBit
Stop bits that signal the end of a character. The entry for the StopBit field can be 1 or 2.

Parity
Checks whether a character was successfully transmitted to or from a remote system. The entry for the Parity field can be 0 (none), 1 (odd), or 2 (even).

Echo
Determines whether typed characters display locally. The entry for the Echo field can be 0 (off) or 1 (on).

Linefeed
Adds a line-feed character at the end of each line of data coming in from a remote system. The line-feed character is similar in function to the carriage-return and new-line characters. The entry for the Linefeed field can be 0 (off) or 1 (on).
Examples
Following is a sample dialing directory entry:

CompuAid  111-0000  1200  7 1 2 0 0

In this example, CompuAid is the Name, 111-0000 is the Phone, 1200 is the Rate, 7 is the Length, 1 is the StopBit, 2 is the Parity, the first 0 is the Echo, and the second 0 is the Linefeed.

Files
ate.def Contains ATE default values.
/usr/lib/dir Contains the default dialing directory listing.

Related Information
The ate command.

The connect subcommand, directory subcommand.

Asynchronous Terminal Emulation Asynchronous communications ate.def configuration file Setting up an ATE dialing directory in Networks and communication management.

DOMAIN Cache File Format for TCP/IP

Purpose
Defines the root name server or servers for a DOMAIN name server host.

Description
The cache file is one of the DOMAIN data files and contains the addresses of the servers that are authoritative name servers for the root domain of the network. The name of this file is defined in the named boot file. If the host serves more than one domain, the cache file should contain an entry for the authoritative name server for each domain.

All entries in this file must be in Standard Resource Record Format. Valid resource records in this file are:
- Name Server (NS)
- Address (A)

Except for comments (starting with a ; [semicolon] and continuing to the end of the line), the resource records in the data files generally follow the format of the resource records that the named daemon returns in response to queries from resolver routines.

Examples
The following examples show the various ways to use the cache data file. This example is valid for any name server or either of the two networks.

Network abc consists of:
- gobi.abc, the primary name server for the abc network, 192.9.201.2
- mojave.abc, a host machine, 192.9.201.6
- sandy.abc, secondary name server for the abc network and gateway between abc and xyz, 192.9.201.3

Network xyz consists of:
- kalahari.xyz, primary name server for the xyz network, 160.9.201.4
• lopnor.xyz, a host machine, 160.9.201.5
• sahara.xyz, a host machine and cache-only name server for the xyz network, 160.9.201.13
• sandy.xyz, a secondary name server for the xyz network and gateway between abc and xyz, 160.9.201.3

Note: sandy, a gateway host, is on both networks and also serves as secondary name server for both.

The following are sample entries in a DOMAIN cache file on any of the name servers in either of the domains:

; cache file for all nameservers in both domains
; root name servers.
abc IN NS gobi.abc.
xyz IN NS kalahari.xyz.
gobi.abc. 3600000 IN A 192.9.201.2
kalahari.xyz 3600000 IN A 100.9.201.4

Files
/etc/named.conf Defines how the named daemon initializes the DOMAIN name server file.
/usr/samples/tcpip/named.conf Sample named.conf file, which also contains directions for its use.
/usr/samples/tcpip/named.data Sample named.data file, which also contains directions for its use.

Related Information
The named daemon.
The DOMAIN Data file format, DOMAIN Reverse Data file format, DOMAIN Local file format.
TCP/IP name resolution and Name server resolution in Networks and communication management.

DOMAIN Data File Format for TCP/IP

Purpose
Stores name resolution information for the named daemon.

Description
The host’s data file is one of the DOMAIN data files and contains name-to-address resolution mapping information for all machines in the name server’s zone of authority. The name of the host’s data file is specified in the named boot file. This file should exist only on name servers that are designated as primary for a domain. There may be more than one host’s data file per primary name server.

All entries in this file must be in Standard Resource Record Format. Valid resource records in this file are:
• Start of Authority (SOA)
• Name Server (NS)
• Address (A)
• Mailbox (MB)
• Mail Exchanger (MX)
• Mail Group (MG)
Except for comments (starting with a ; (semicolon) and continuing to the end of the line), the resource records in the data files generally follow the format of the resource records that the named daemon returns in response to queries from resolver routines.

Two awk scripts, addrs.awk and hosts.awk, are provided in the /usr/samples/tcpip directory to assist you in converting your existing /etc/hosts file to DOMAIN data files. The awk scripts also contain instructions for their use. Refer to these files for more information on the conversion.

Examples
The following examples show the various ways to use the DOMAIN host's data file. In these examples, two networks are represented: abc and xyz.

Network abc consists of:
- gobi.abc, the primary name server for the abc network, 192.9.201.2
- mojave.abc, a host machine, 192.9.201.6
- sandy.abc, secondary name server for the abc network and gateway between abc and xyz, 192.9.201.3

Network xyz consists of:
- kalahari.xyz, primary name server for the xyz network, 160.9.201.4
- lopnor.xyz, a host machine, 160.9.201.5
- sahara.xyz, a host machine and cache-only name server for the xyz network, 160.9.201.13
- sandy.xyz, a secondary name server for the xyz network and gateway between abc and xyz, 160.9.201.3

Note: Host sandy, a gateway host, is on both networks and also serves as secondary name server for both.

1. The primary host data file for network abc, stored on host gobi.abc, contains the following entries:

```plaintext
@     IN  SOA      gobi.abc.  root.gobi.abc.  (  
  1.1 ;serial
  3600 ;refresh
  600  ;retry
  3600000;expire
  86400 ;minimum
 )

;name servers for abc
IN      NS       gobi.abc.

;other name servers
IN      NS       kalahari.xyz.

kalahari.xyz. IN      A          160.9.201.4

;define local loopback host
localhost IN      A          127.1

;define all hosts in abc
loopback IN      CNAME     localhost.abc

   gobi     IN      A          192.9.201.2
   gobi-abc IN      CNAME     gobi.abc
   sandy    IN      A          192.9.201.3
       IN      WKS     192.9.201.3
```
udp  tftp  nameserver  domain  
  IN    WKS    192.9.201.3  tcp  {  
    echo  telnet  smtp  discard  uucp-path  
    systat  daytime  netstat  chargen  ftp  
    time  whois  finger  hostnames  domain  
  }  

sandy-abc  IN    CNAME    sandy.abc  
mojave  IN    A    192.9.201.6  
mojave-abc  IN    HINFO    System  ABC  3.1  

2. The primary host data file for network xyz, stored on host kalahari.xyz, contains the following entries:  

```plaintext 
@  IN    SOA    kalahari.xyz. root.kalahari.xyz. ( 
  1.1  ;serial  
  3600  ;refresh  
  600  ;retry  
  3600000;expire  
  86400  ;minimum  
)

;  nameservers for xyz  
;  IN    NS    kalahari.xyz.  
;  other nameservers  
gobi.abc.  IN    A    192.9.201.2  

;  define local loopback host  
localhost  IN    A    127.1  

;  define all hosts in xyz  
loopback  IN    CNAME    localhost.xyz.  
kalahari  IN    A    160.9.201.4  
ns-xyz  IN    CNAME    kalahari.xyz.  
kalahari-xyz  IN    CNAME    kalahari.xyz.  
IN    HINFO    System  ABC  3.1  
sahara  IN    A    160.9.201.13  
IN    WKS    160.9.201.13  (  
  udp  tftp  nameserver  domain  )  
IN    WKS    160.9.201.13  tcp  {  
    echo  telnet  smtp  discard  uucp-path  
    systat  daytime  netstat  chargen  ftp  
    time  whois  finger  hostnames  domain  
  }  
IN    HINFO    System  ABC  3.1  

lopnor  IN    A    160.9.201.5  
lopnor-xyz  IN    CNAME    lopnor.xyz.  
IN    HINFO    System  ABC  3.1  
sandy  IN    A    160.9.201.3  
```
Related Information

The named daemon.

The DOMAIN Reverse Data file format, DOMAIN Cache file format, DOMAIN Local file format.

Standard Resource Record Format for TCP/IP

TCP/IP name resolution and Name server resolution in Networks and communication management.

DOMAIN Local Data File Format for TCP/IP

Purpose

Defines the local loopback information for the named daemon on the name server host.

Description

The local data file is one of the DOMAIN data files and contains local loopback information for the name-server host. The name of the DOMAIN local data files is specified in the named boot file.

All entries in this file must be in Standard Resource Record Format. Valid resource records in the local data file are:

- Start of Authority (SOA)
- Name Server (NS)
- Pointer (PTR)

The records in the DOMAIN data files are called resource records. Except for comments (starting with a ; (semicolon) and continuing to the end of the line), the resource records in the data files generally follow the format of the resource records that the named daemon returns in response to queries from resolver routines.

Examples

The following examples show the various ways to use the DOMAIN local data file. In these examples, two networks are represented: abc and xyz.

Network abc consists of:

- gobi.abc, the primary name server for the abc network, 192.9.201.2
- mojave.abc, a host machine, 192.9.201.6
- sandy.abc, secondary name server for the abc network and gateway between abc and xyz, 192.9.201.3.

Network xyz consists of:

- kalahari.xyz, primary name server for the xyz network, 160.9.201.4
- lopnor.xyz, a host machine, 160.9.201.5
- sahara.xyz, a host machine and cache-only name server for the xyz network, 160.9.201.13
- sandy.xyz, a secondary name server for the xyz network and gateway between abc and xyz, 160.9.201.3
Note: Host sandy, a gateway host, is on both networks and also serves as secondary name server for both.

1. The **named.abclocal** file stored on gobi.abc contains the following entries:

```
; primary reverse file for local 127 network
@ IN SOA gobi.abc. root.gobi.abc.
  ( 1.1 ;serial
  3600 ;refresh
  600 ;retry
  3600000;expire
  86400 ;minimum
  )
IN NS gobi.abc.
1 IN PTR localhost.
```

2. The **named.xyzlocal** file stored on kalahari.xyz contains the following entries:

```
; primary reverse file for local 127 network
@ IN SOA kalahari.xyz. root.kalahari.xyz.
  ( 1.1 ;serial
  3600 ;refresh
  600 ;retry
  3600000;expire
  86400 ;minimum
  )
IN NS kalahari.xyz.
1 IN PTR localhost.
```

3. The **named.seclocal** file stored on sandy contains the following entries:

```
; primary reverse file for local 127 network
@ IN SOA sandy.abc. root.sandy.abc.
  ( 1.1 ;serial
  3600 ;refresh
  600 ;retry
  3600000;expire
  86400 ;minimum
  )
IN NS sandy.abc.
1 IN PTR localhost.
```

4. The **named.calocal** file stored on sahara.xyz contains the following entries:

```
; primary reverse file for local 127 network
@ IN SOA sahara.xyz. root.sahara.xyz.
  ( 1.1 ;serial
  3600 ;refresh
  600 ;retry
  3600000;expire
  86400 ;minimum
  )
IN NS sahara.xyz.
1 IN PTR localhost.
```
Files

/etc/named.conf
Defines how the named daemon initializes the DOMAIN name-server file.

/usr/samples/tcpip/named.conf
Sample named.conf file, which also contains directions for its use.

/usr/samples/tcpip/named.data
Sample named.data file, which also contains directions for its use.

Related Information

The named daemon.

The DOMAIN Data file format, DOMAIN Reverse Data file format, DOMAIN Cache file format.

TCP/IP name resolution and Name server resolution in Networks and communication management.

DOMAIN Reverse Data File Format for TCP/IP

Purpose
Stores reverse name resolution information for the named daemon.

Description
The Reverse Data file is one of the DOMAIN data files and contains address to name resolution mapping information for all machines in the name server's zone of authority. The name of the reverse hosts data file is specified in the named boot file. There may be more than one reverse hosts data file per primary name server.

All entries in this file must be in Standard Resource Record Format. Valid resource records in this file are:
- Start of Authority (SOA)
- Name Server (NS)
- Pointer (PTR)

Except for comments (starting with a ; (semicolon) and continuing to the end of the line), the resource records in the data files generally follow the format of the resource records that the named daemon returns in response to queries from resolver routines.

Two awk scripts, addrs.awk and hosts.awk, are provided in the /usr/samples/tcpip directory to assist you in converting your existing /etc/hosts file to named data files. The awk scripts also contain instructions for their use. Refer to these files for more information on the conversion.

Examples
The following examples show the various ways to use the DOMAIN Reverse Data file. In these examples, two networks are represented: abc and xyz.

Network abc consists of:
- gobi.abc, the primary name server for the abc network, 192.9.201.2
- mojave.abc, a host machine, 192.9.201.6
- sandy.abc, secondary name server for the abc network and gateway between abc and xyz, 192.9.201.3

Network xyz consists of:
- kalahari.xyz, primary name server for the xyz network, 160.9.201.4
- lopnor.xyz, a host machine and cache-only name server for the xyz network, 160.9.201.5
- sahara.xyz, a host machine, 160.9.201.13
- sandy.xyz, a secondary name server for the xyz network and gateway between abc and xyz, 160.9.201.3

**Note:** Host sandy, a gateway host, is on both networks and also serves as secondary name server for both.

1. The reverse data file for gobi.abc, primary name server for network abc, contains these entries:

   ```
   @           IN      SOA      gobi.abc.   root.gobi.abc. ( 
     1:1 ;serial 
     3600 ;refresh 
     600 ;retry  
     3600000;expire 
     86400 ;minimum  
   )
   
   ;nameservers for abc
   IN      NS      gobi.abc.
   ;other nameservers
   IN      NS      kalahari.xyz.
   4.201.9.160.in-addr.arpa   IN      PTR      kalahari.xyz
   
   ;define all hosts in abc
   2      IN      PTR      gobi.abc.
   3      IN      PTR      sandy.abc.
   6      IN      PTR      mojave.abc.
   ```

2. The reverse data file for kalahari.xyz, primary name server for network xyz, contains these entries:

   ```
   @           IN      SOA      kalahari.xyz.   root.kalahari.xyz. ( 
     1:1 ;serial 
     3600 ;refresh 
     600 ;retry 
     3600000;expire 
     86400 ;minimum  
   )
   
   ;nameservers for xyz
   IN      NS      kalahari.xyz.
   ;other nameservers
   IN      NS      gobi.abc.
   2.201.9.192.in-addr.arpa   IN      PTR      gobi.abc
   
   ;define all hosts in xyz
   4.201      IN      PTR      kalahari.xyz.
   13.201      IN      PTR      sahara.xyz.
   5.201      IN      PTR      lopnor.xyz.
   3.201      IN      PTR      sandy.xyz.
   ```

**Files**

- `/etc/named.conf`
  - Defines how the `named` daemon initializes the DOMAIN name server file.

- `/usr/samples/tcpip/addrs.awk`
  - Sample `awk` script for converting an `/etc/hosts` file to an `/etc/named.rev` file. The `awk` script also contains directions for its use.

- `/usr/samples/tcpip/hosts.awk`
  - Sample `awk` script for converting an `/etc/hosts` file to an `/etc/named.data` file. The `awk` script also contains directions for its use.
/usr/samples/tcpip/named.conf
Contains a sample named.conf file, which also contains directions for its use.

/usr/samples/tcpip/named.data
Contains a sample named.data file, which also contains directions for its use.

Related Information
The named daemon.


Standard Resource Record Format for TCP/IP
TCP/IP name resolution and Name server resolution in Networks and communication management.

eqnchar File Format

Purpose
Contains special character definitions for the eqn and neqn commands.

Description
The /usr/share/lib/pub/eqnchar file contains the following troff and nroff command character definitions not ordinarily available on a phototypesetter or printer. These definitions are primarily intended for use with the eqn and neqn commands. The eqnchar file format contains definitions for the characters shown in the following character definition list.

```
ciplus          ⊕      ||          ||      square          □
citimes         ⊗      ]|           ]|      circle           ○
wig             ¬  
-wig            ¬  
>wig            ↑      rangle          (          bullet           •
<wig            ↓      rangle          )          prop            ♀
-wig            ↓      hbar            h          empty             Ø
star            *      <            <        nonmem             ±
bigstar         *      >            >        cup              ♀
-dot            ¦      ang            <          cap              ♀
orSIGN           ∨      rang            L          incl              ☐
andSIGN         ∧      3dot          ::          subset            ☐
-wdel           △      thf            ::          superset           ☐
oppA            ∀      quarter      1/4        !subset           ☐
oppE            ∃      3quarter     3/4        !superset          ☐
angstrom       Å      degree       °          scrL             ℓ
```

This illustration shows symbols commonly used in equations.
The `/usr/share/lib/pub/cateqnchar` file is device-independent and should produce output that looks reasonable on any device supported by the `troff` command. You can link the `/usr/share/lib/pub/eqnchar` file to the `/usr/share/lib/pub/cateqnchar` file.

The `eqnchar` file format can be used with either the `eqn` or `neqn` command and then piped to the `troff` or `nroff` command. For example:

```plaintext
eqn /usr/share/lib/pub/eqnchar [ Flag... ] [ — ] [ File... ] l troff [ Flag... ]
eqn /usr/share/lib/pub/cateqnchar [ Flag... ] [ — ] [ File... ] l troff [ Flag... ]
neqn /usr/share/lib/pub/eqnchar [ Flag... ] [ — ] [ File... ] l nroff [ Flag... ]
```

**Files**

`/usr/share/lib/pub/cateqnchar` Contains the character definitions for `troff`-supported device.

**Related Information**

The `eqn` command, `mm` command, `mmt` command, `mvt` command, `neqn` command, `nroff` command, `troff` command.

### ftpusers File Format for TCP/IP

**Purpose**

Specifies local user names that cannot be used by remote FTP clients.

**Description**

The `/etc/ftpusers` file contains a list of local user names that the `ftpd` server does not allow remote File Transfer Protocol (FTP) clients to use. The format of the `ftpusers` file is a simple list of user names that also appear in the `/etc/passwd` file.

Entries to this file can be made using the System Management Interface Tool (SMIT) or the `ruser` command.

**Examples**

The following are sample entries in an `ftpusers` file:

```
root
guest
ftp
joan
UUCP
```

**Files**

`/etc/passwd` Contains user authentication information.

**Related Information**

The `ruser` command.

The `ftpd` daemon.
gated.conf File Format for TCP/IP

Purpose
Contains configuration information for the gated daemon.

Description
The /etc/gated.conf file contains configuration information for the gated daemon. The file contains a sequence of statements. Statements are composed of tokens separated by white space. You can create white space using any combination of blanks, tabs, and new lines. The gated.conf file supports several statements:

- `%directory` (directive) Sets the directory for include files.
- `%include` (directive) Includes a file into gated.conf.
- `traceoptions` (trace) Specifies which events are traced.
- `options` (definition) Defines gated.conf options.
- `interfaces` (definition) Defines gated.conf interfaces.
- `autonomoussystem` Defines the AS number.
- `routerid` (definition) Defines the originating router (BGP, OSPF).
- `martians` (definition) Defines invalid destination addresses.
- `rip` (protocol) Enables RIP protocol.
- `ripng` Enables or disables RIPNG. If the RIPNG statement is not specified, the default is ripng on ;. The options are the same for RIPNG as they are for RIP, but all the addresses will be IPv6 addresses.
- `hello` (protocol) Enables HELLO protocol.
- `isis` (protocol) Enables ISIS protocol.
- `ospf` (protocol) Enables OSPF protocol.
- `EGP` (protocol) Enables EGP protocol.
- `bgp` (protocol) Enables BGP protocol.
- `bgp4+` The options are the same as bgp but all the addresses will be IPv6 addresses.
- `icmp` (protocol) Configures the processing of general ICMP packets.
- `snmp` (protocol) Enables reporting to SNMP.
- `static` (static) Defines static routes.
- `import` (control) Defines which routes to import.
- `export` (control) Defines which routes to export.
- `aggregate` (control) Defines which routes to aggregate.
- `generate` (control) Defines which routes to generate.

Directive Statements
Directive statements provide direction to the gated.conf configuration language parser about included files and the directories in which these files reside. Directive statements are immediately acted upon by the parser. Other statements terminate with a semi-colon (;), but directive statements terminate with a newline. The two directive statements are:

- `%directory "directory"` Defines the directory where the include files are stored. When it is used, gated.conf looks in the directory identified by pathname for any included files that do not have a fully qualified filename, that is, do not begin with "/". This statement does not actually change the current directory, it just specifies the prefix applied to included file names.
%include “filename”

Identifies an include file. The contents of the file are included in the gated.conf file at the point in the gated.conf file where the %include directive is encountered. If the filename is not fully qualified, that is, does not begin with “/”, it is considered to be relative to the directory defined in the %directory directive. The %include directive statement causes the specified file to be parsed completely before resuming with this file. Nesting up to ten levels is supported.

In a complex environment, segmenting a large configuration into smaller more easily understood segments might be helpful, but one of the great advantages of gated.conf is that it combines the configuration of several different routing protocols into a single file. Segmenting a small file unnecessarily complicates routing configurations.

Trace Statements
Trace statements control tracing options. gated.conf’s tracing options may be configured at many levels. Tracing options include the file specifications, control options, and global and protocol specific tracing options. Unless overridden, tracing options from the next higher level are inherited by lower levels. For example, BGP peer tracing options are inherited from BGP group tracing options, which are inherited from global BGP tracing options, which are inherited from global gated.conf tracing options. At each level, tracing specifications override the inherited options.

Global tracing options
There are two types of global options, those that only affect global operations, and those that have potential significance to protocols.

Global significance only
The trace flags that only have global significance are:

- parse: Traces the lexical analyzer and parser. Mostly used by gated.conf developers for debugging.
- adv: Traces the allocation of and freeing of policy blocks. Mostly used by the gated.conf developers for debugging.
- symbols: Used to trace symbols read from the kernel at startup. The only useful way to specify this level of tracing is via the -t option on the command line since the symbols are read from the kernel before parsing the configuration file.
- iflist: Used to trace the reading of the kernel interface list. It is useful to specify this with the -t option on the command line since the first interface scan is done before reading the configuration file.

Protocol significance
The options flags that have potential significance to protocols are:

- all: Turn on all of the following.
- general: Shorthand notation for specifying both normal and route.
- state: Trace state machine transitions in the protocols.
- normal: Trace normal protocols occurrences. Abnormal protocol occurrences are always traced.
- policy: Trace application of protocol and user-specified policy to routes being imported and exported.
- task: Trace system interface and processing associated with this protocol or peer.
- timer: Trace timer usage by this protocol or peer.
- route: Trace routing table changes for routes installed by this protocol or peer.

Notes:
1. Not all of the above options apply to all of the protocols. In some cases, their use does not make sense (for instance, RIP does not have a state machine) and in some instances the requested tracing has not been implemented (such as RIP support of the policy option).
2. It is not currently possible to specify packet tracing from the command line. This is because a global option for packet tracing would potentially create too much output.

When protocols inherit their tracing options from the global tracing options, tracing levels that don’t make sense (such as parse, adv and packet tracing options) are masked out.

Global tracing statements have an immediate effect, especially parsing options that affect the parsing of the configuration file. Tracing values inherited by protocols specified in the configuration file are initially inherited from the global options in effect as they are parsed, unless they are overridden by more specific options. After the configuration file is read, tracing options that were not explicitly specified are inherited from the global options in effect at the end of the configuration file.

Packet tracing
Tracing of packets is very flexible. For any given protocol, there are one or more options for tracing packets. All protocols allow use of the packets keyword that allows for tracing all packets sent and received by the protocol. Most protocols have other options for limiting tracing to a useful subset of packet types. These tracing options can be further controlled with the following modifiers:

- **detail**: The detail must be specified before send or recv. Normally packets are traced in a terse form of one or two lines. When detail is specified, a more verbose format is used to provide further detail on the contents of the packet.

- **send**: These options limit the tracing to packets sent or received. Without these options both sent and received packets will be traced.

- **recv**: These options limit the tracing to packets sent or received. Without these options both sent and received packets will be traced.

**Note**: Detail, if specified, must be before send or recv. If a protocol allows for several different types of packet tracing, modifiers may be applied to each individual type. But be aware that within one tracing specification the trace flags are summed up, so specifying detail packets will turn on full tracing for all packets.

Traceoptions syntax

```
traceoptions ["trace_file" [replace] [size size[k|m] files files]]
[control_options] trace_options [except trace_options];
```

- **trace_file**: Specifies the file to receive tracing information. If this file name does not begin with a slash (/), the directory where gated was started is prepended to the name.

- **replace**: Indicates tracing should start by replacing an existing file. The default is to append to an existing file.

- **size size[k|m] files files**: Limits the maximum size of the trace file to the specified size (minimum 10k). When the trace file reaches the specified size, it is renamed to file.0, then file.1, file.2 up to the maximum number of files (minimum specification is 2).

- **control_options**: Specifies options that control the appearance of tracing. Valid values are:

  - **nostamp**: Specifies that a timestamp should not be prepended to all trace lines.

  - **except trace_options**: Used to enable a broad class of tracing and then disable more specific options.

  - **none**: Specifies that all tracing should be turned off for this protocol or peer.
Options Statements

Options statements allow specification of some global options. If used, options must appear before any other type of configuration statement in the gated.conf file.

The options statement syntax is:

```
options
  [nosend ]
  [noresolv ]
  [gendefault [preference preference] [gateway gateway] ]
  [syslog [upto ] log_level ]
  [mark time ]
```

The options list can contain one or more of the following options:

- **gendefault [preference preference] [gateway gateway]**: When gendefault is enabled when a BGP or EGP neighbor is up, it causes the creation of a default route with the special protocol default. This can be disabled per BGP/EGP group with the nogendefault option. By default, this route has a preference of 20. This route is normally not installed in the kernel forwarding table, it is only present so it can be announced to other protocols. If a gateway is specified, the default route will be installed in the kernel forwarding table with a next hop of the listed gateway.
  
  **Note:** The use of the more general generate default option is preferred to the use of this gendefault option. See the section on [Route Aggregation](#) for more information on the generate statement.

- **nosend**: Do not send any packets. This option makes it possible to run gated.conf on a live network to test protocol interactions without actually participating in the routing protocols. The packet traces in the gated.conf log can be examined to verify that gated.conf is functioning properly. This is most useful for RIP and HELLO.

- **noresolv**: By default, gated.conf will try to resolve symbolic names into IP addresses; this option will prevent that.

- **syslog [upto ] log_level**: Controls the amount of data gated.conf logs via syslog.

- **mark time**: Specifying this option causes gated to output a message to the trace log at the specified interval. This can be used as one method of determining if gated is still running.

Interface Statement

Interface Syntax

```
interfaces {
  options
    [ strictinterfaces ]
    [ scaninterval time ]
  ;
  interface interface_list
    [ preference preference ]
    [ down preference preference ]
    [ passive ]
    [ simplex ]
    [ reject ]
    [ blackhole ]
  ;
  define address
    [ broadcast address ] | [ pointtopoint address ]
```


An interface is the connection between a router and one of its attached networks. A physical interface may be specified by interface name, by IP address, or by domain name, (unless the network is an unnumbered point-to-point network.) Multiple levels of reference in the configuration language allow identification of interfaces using wildcard, interface type name, or delete word addresses. The interface_list is a list of one or more interface names including wildcard names (names without a number) and names that may specify more than one interface or address, or the token all for all interfaces.

interface interface_list

Sets interface options on the specified interfaces. An interface list is all or a list of interface names domain names, or numeric addresses. Options available on this statement are:

preference preference
Sets the preference for routes to this interface when it is up and appears to be functioning properly. The default preference is 0.

down preference preference
Sets the preference for routes to this interface when the gated daemon does not believe it to be functioning properly, but the kernel does not indicate it is down. The default value is 120.

passive
Prevents the gated daemon from changing the preference of the route to this interface if it is not believed to be functioning properly due to lack of received routing information. The gated daemon will only perform this check if the interface is actively participating in a routing protocol.

define address
Defines interfaces that might not be present when the gated daemon is started so they may be referenced in the configuration file when strictinterfaces is defined. Possible define keywords are:

broadcast address
Defines the interface as broadcast capable (for example, Ethernet or Token Ring) and specifies the broadcast address.

pointopoint address
Defines the interface as a pointopoint interface (for example, SLIP or PPP) and specifies the address on the local side. The first address on the define statement references the address of the host on the remote end of the interface, the address specified after this pointopoint keyword defines the address on the local side of the interface.

An interface not defined as broadcast or pointopoint is assumed to be non-broadcast multiaccess (NBMA), such as an X.25 network.

netmask mask
Specifies the subnetmask to be used on this interface. This is ignored on pointopoint interfaces.

multicast
Specifies that the interface is multicast capable.

Interface Lists
An interface list is a list of references to interfaces or groups of interfaces. There are four methods available for referring to interfaces. They are listed here from most general to most specific.

call
This refers to all available interfaces.
Interface name wildcard
This refers to all the interfaces of the same type. The operating system interfaces consist of the name of the device driver, like en, and a unit number, like 0 or 5. References to the name contain only alphabetic characters and match any interfaces that have the same alphabetic part. For example, en would refer to all Ethernet interfaces.

Interface name
This refers to a specific interface, usually one physical interface. These are specified as an alphabetic part followed by a numeric part. This will match one specific interface. For example, en1 will match an interface named en1, but not an interface named en10. In case there are multiple addresses aliased to a single interface, specify the particular IP address to be used by gated, instead of the interface name.

Interface address
This matches one specific interface. The reference can be by protocol address (that is, 10.0.0.51), or by symbolic hostname (that is, hornet.ibm.com). Note that a symbolic hostname reference is only valid when it resolves to only one address. Use of symbolic hostnames is not recommended.

If many interface lists are present in the config file with more than one parameter, these parameters are collected at run-time to create the specific parameter list for a given interface. If the same parameter is specified on more than one list, the parameter with the most specific interface is used.

For example, consider a system with three interfaces: en0, en1, and tr0.

```plaintext
rip yes {
    interface all noripin noripout;
    interface en ripin;
    interface en1 ripout;
}
```

RIP packets would only be accepted from interfaces en0 and en1, but not from tr0. RIP packets would only be sent on interface en1.

**IP Interface Addresses and Routes**

**loopback**
This interface must have the address of 127.0.0.1. Packets sent to this interface are sent back to the originator. This interface is also used as a catch-all interface for implementing other features, such as reject and blackhole routes. Although a netmask is reported on this interface, it is ignored. It is useful to assign an additional address to this interface that is the same as the OSPF or BGP router id; this allows routing to a system based on the router id that will work if some interfaces are down.

**broadcast**
This is a multi-access interface capable of a physical level broadcast, such as Ethernet, Token Ring, and FDDI. This interface has an associated subnet mask and broadcast address. The interface route to a broadcast network will be a route to the complete subnet.

**point-to-point**
This is a tunnel to another host, usually on some sort of serial link. This interface has a local address, and a remote address.

The remote address must be unique among all the interface addresses on a given router. The local address may be shared among many point-to-point and up to one non-point-to-point interface. This is technically a form of the router id method for addressless links. This technique conserves subnets as none are required when using this technique.

If a subnet mask is specified on a point-to-point interface, it is only used by RIP version 1 and HELLO to determine which subnets may be propagated to the router on the other side of this interface.
non-broadcast multi-access or nbma

This type of interface is multi-access, but not capable of broadcast. An example would be frame relay and X.25. This type of interface has a local address and a subnet mask.

The gated daemon insures that there is a route available to each IP interface that is configured and up. Normally this is done by the ifconfig command that configures the interface; the gated daemon does it to insure consistency.

For point-to-point interfaces, the gated daemon installs some special routes. If the local address on one or more point-to-point interfaces is not shared with a non-point-to-point interface, the gated daemon installs a route to the local address pointing at the loopback interface with a preference of 110. This insures that packets originating on this host destined for this local address are handled locally. OSPF prefers to route packets for the local interface across the point-to-point link where they will be returned by the router on the remote end. This is used to verify operation of the link. Since OSPF installs routes with a preference of 10, these routes will override the route installed with a preference of 110.

If the local address of one or more point-to-point interfaces is shared with a non-point-to-point interface, the gated daemon installs a route to the local with a preference of 0 that will not be installed in the forwarding table. This is to prevent protocols like OSPF from routing packets to this address across a serial interface when this system could be functioning as a host.

When the status of an interface changes, the gated daemon notifies all the protocols, which take the appropriate action. The gated daemon assumes that interfaces that are not marked up do not exist.

The gated daemon ignores any interfaces that have invalid data for the local, remote, or broadcast addresses or the subnet mask. Invalid data includes zeros in any field. The gated daemon will also ignore any point-to-point interface that has the same local and remote addresses.

**Definition Statements**

Definition statements are general configuration statements that relate to all of gated daemon or at least to more than one protocol. The three definition statements are autonomoussystem, routerid, and martians. If used, autonomoussystem, routerid, and martians must appear before any other type of configuration statement in the gated daemon file.

### Autonomous System Configuration

```
autonomoussystem autonomous_system [loops number] ;
```

Sets the autonomous system number of this router to be autonomous system. This option is required if BGP or EGP are in use. The AS number is assigned by the Network Information Center (NIC).

Loops is only for protocols supporting AS paths, such as BGP. It controls the number of times this autonomous system may appear in an AS path and defaults to 1 (one).

### Router ID Configuration

```
routerid host ;
```

Sets the router identifier for use by the BGP and OSPF protocols. The default is the address of the first interface encountered by the gated daemon. The address of a non-point-to-point interface is preferred over the local address of a point-to-point interface and an address on a loopback interface that is not the loopback address (127.0.0.1) is most preferred.
Martian Configuration

```plaintext
martians {
    host host [allow] ;
    network [allow] ;
    network mask mask [allow] ;
    network masklen number [allow] ;
    default [allow] ;
}
```

Defines a list of martian addresses about which all routing information is ignored. Sometimes a misconfigured system sends out obviously invalid destination addresses. These invalid addresses, called martians, are rejected by the routing software. This command allows additions to the list of martian addresses. See the section on Route Filtering for more information on specifying ranges. Also, the allow parameter may be specified to explicitly allow a subset of a range that was disallowed.

Sample Definition Statements

```plaintext
options gendefault ;
autonomoussystem 249 ;
interface 128.66.12.2 passive ;
martians {
    0.0.0.26
}
```

The statements in the sample perform the following functions:

- The options statement tells the system to generate a default route when it peers with an EGP or BGP neighbor.
- The autonomoussystem statement tells the `gated` daemon to use the AS number 249 for EGP and BGP.
- The interface statement tells the `gated` daemon not to mark interface 128.66.12.2 as down even if it sees no traffic.
- The martians statement prevents routes to 0.0.0.26 from ever being accepted.

The RIP Statement

```plaintext
rip yes | no | on | off [{
    broadcast ;
    nobroadcast ;
    nocheckzero ;
    preference preference ;
    defaultmetric metric ;
    query authentication [none | [simple|md5] password]] ;
    interface [interface_list]
        [noripin] | [ripin]
        [noripout] | [ripout]
        [metricin metric]
        [metricout metric]
        [version 1][version 2 [multicast|broadcast]]
        [[secondary] authentication [none | [[simple|md5] password]] ;
    trustedgateways gateway_list ;
    sourcegateways gateway_list ;
    traceoptions trace_options ;
}
```

The `rip` statement enables or disables RIP. If the `rip` statement is not specified, the default is `rip on`. If enabled, RIP will assume `nobroadcast` when there is only one interface and `broadcast` when there is more than one.
The options are as follows:

**broadcast**
Specifies that RIP packets will be broadcast regardless of the number of interfaces present. This is useful when propagating static routes or routes learned from another protocol into RIP. In some cases, the use of broadcast when only one network interface is present can cause data packets to traverse a single network twice.

**nobroadcast**
Specifies that RIP packets will not be broadcast on attached interfaces, even if there is more than one. If a sourcegateways clause is present, routes will still be unicast directly to that gateway.

**nocheckzero**
Specifies that RIP should not make sure that reserved fields in incoming version 1 RIP packets are zero. Normally RIP will reject packets where the reserved fields are zero.

**preference**
Sets the preference for routes learned from RIP. The default preference is 100. This preference may be overridden by a preference specified in import policy.

**defaultmetric**
Defines the metric used when advertising routes via RIP were learned from other protocols. If not specified, the default value is 16 (unreachable). This choice of values requires you to explicitly specify a metric in order to export routes from other protocols into RIP. This metric may be overridden by a metric specified in export policy.

**query authentication** [none | [simple|md5] password]
Specifies the authentication required of query packets that do not originate from routers. The default is none.

**interface** [interface_list]
Controls various attributes of sending RIP on specific interfaces. See the section on interface list specification for a description of the [interface_list].

**Note:** If there are multiple interfaces configured on the same subnet, RIP updates will only be sent from the first one from which RIP output is configured.

The possible parameters are:

**noripin**
Specifies that RIP packets received via the specified interface will be ignored. The default is to listen to RIP packets on all non-loopback interfaces.

**ripin**
This is the default. This argument may be necessary when noripin is used on a wildcard interface descriptor.

**noripout**
Specifies that no RIP packets will be sent on the specified interfaces. The default is to send RIP on all broadcast and non-broadcast interfaces when in broadcast mode. The sending of RIP on point-to-point interfaces must be manually configured.

**ripout**
This is the default. This argument is necessary when it is desired to send RIP on point-to-point interfaces and may be necessary when noripin is used on a wildcard interface descriptor.

**metricin** [metric]
Specifies the RIP metric to add to incoming routes before they are installed in the routing table. The default is the kernel interface metric plus 1 (which is the default RIP hop count). If this value is specified it will be used as the absolute value, the kernel metric will not be added. This option is used to make this router prefer RIP routes learned via the specified interface(s) less than RIP routes from other interfaces.
metricout metric
Specifies the RIP metric to be added to routes that are sent via the specified interface(s).
The default is zero. This option is used to make other routers prefer other sources of RIP
routes over this router.

version 1
Specifies that RIP packets sent via the specified interface(s) will be version 1 packets.
This is the default.

version 2
Specifies that RIP version 2 packets will be sent on the specified interfaces(s). If IP
multicast support is available on this interface, the default is to send full version 2 packets.
If it is not available, version 1 compatible version 2 packets will be sent.

multicast
Specifies that RIP version 2 packets should be multicast on this interface. This is the
default.

broadcast
Specifies that RIP version 1 compatible version 2 packets should be broadcast on this
interface, even if IP multicast is available.

[secondary] authentication [none l [simple|md5] password]
This defines the authentication type to use. It applies only to RIP version 2 and is ignored
for RIP-1 packets. The default authentication type is none. If a password is specified, the
authentication type defaults to simple. The password should be a quoted string with
between 0 and 16 characters.

If secondary is specified, this defines the secondary authentication. If omitted, the primary
authentication is specified. The default is primary authentication of none and no secondary
authentication.

trustedgateways gateway_list
Defines the list of gateways from which RIP will accept updates. The gateway_list is simply a list
of host names or IP addresses. By default, all routers on the shared network are trusted to supply
routing information. But if the trustedgateways clause is specified, only updates from the gateways
in the list are accepted.

sourcegateways gateway_list
Defines a list of routers to which RIP sends packets directly, not through multicast or broadcast.
This can be used to send different routing information to specific gateways. Updates to gateways
in this list are not affected by noripout on the interface.

traceoptions trace_options
Specifies the tracing options for RIP. (See Trace Statements and the RIP specific tracing options
below.)

Tracing options
The policy option logs info whenever a new route is announced, the metric being announced changes, or
a route goes or leaves holddown.

Packet tracing options (which may be modified with detail, send, or recv):

packets All RIP packets.
request All RIP information request packets, such as REQUEST, POLL, and POLLENTRY.
response All RIP RESPONSE packets, which are the types of packets that actually contains routing
information.
other Any other type of packet. The only valid ones are TRACE_ON and TRACE_OFF both of which are
ignored.
The RIPNG Statement

Enables or disables ripng. If the ripng statement is not specified, the default is ripng on ;. The options are the same as for rip, but all the addresses will be IPv6 addresses.

The syntax is:

```
ripng yes | no | on | off [ { 
  broadcast ;
  nobroadcast ;
  nocheckzero ;
  preference <preference> ;
  defaultmetric <metric> ;
  query authentication [none | [[simple|md5] <password>]] ;
interface <interface_list>
  [noripin] | [ripin]
  [noripout] | [ripout]
  [metricin <metric>]
  [metricout <metric>]
  [version 1] | [version 2] [multicast|broadcast]
  [[secondary] authentication [none | [[simple|md5] <password>]] ;
trustgateways <gateway_list> ;
sourcegateways <gateway_list> ;
traceoptions <trace_options> ;
} ] ;
```

The Hello Statement

```
hello yes | no | on | off [{
  broadcast ;
  nobroadcast ;
  preference <preference> ;
  defaultmetric <metric> ;
interface <interface_list>
  [nohelloin] | [helloin]
  [nohelloout] | [helloout]
  [metricin <metric>]
  [metricout <metric>]
  trustedgateways <gateway_list> ;
sourcegateways <gateway_list> ;
traceoptions <trace_options> ;
} ] ;
```

The hello statement enables or disables HELLO. If the hello statement is not specified, the default is hello off. If enabled, HELLO will assume nobroadcast when there is only one interface and broadcast when there is more than one interface.

- **broadcast**
  Specifies that HELLO packets will be broadcast regardless of the number of interfaces present. This is useful when propagating static routes or routes learned from another protocol into HELLO. In some cases, the use of broadcast when only one network interface is present can cause data packets to traverse a single network twice.

- **nobroadcast**
  Specifies that HELLO packets will not be broadcast on attached interfaces, even if there are more than one. If a sourcegateways clause is present, routes will still be unicast directly to that gateway.

- **preference**
  Sets the preference for routes learned from HELLO. The default preference is op. This preference may be overridden by a preference specified in import policy.
defaultmetric metric

Defines the metric used when advertising routes via HELLO were learned from other protocols. If not specified, the default value is 30000 (unreachable). This choice of values requires you to explicitly specify a metric in order to export routes from other protocols into HELLO. This metric may be overridden by a metric specified in export policy.

interface interface_list

Controls various attributes of sending HELLO on specific interfaces. See the section on interface list specification for the description of the interface_list.

Note: If there are multiple interfaces configured on the same subnet, HELLO updates will only be sent from the first one from which the HELLO output is configured.

The possible parameters are:

nohelloin

Specifies that HELLO packets received via the specified interface will be ignored. The default is to listen to HELLO on all non-loopback interfaces.

helloin

This is the default. This argument may be necessary when nohelloin is used on a wildcard interface descriptor.

nohelloworldout

Specifies that no HELLO packets will be sent on the specified interfaces. The default is to send HELLO on all broadcast and non-broadcast interfaces when in broadcast mode. The sending of HELLO on point-to-point interfaces must be manually configured.

helloout

This is the default. This argument is necessary when it is desired to send HELLO on point-to-point interfaces and may be necessary when nohelloin is used on a wildcard interface descriptor.

metricin metric

Specifies the HELLO metric to add to incoming routes before they are installed in the routing table. The default is the kernel interface metric plus 1 (which is the default HELLO hop count). If this value is specified it will be used as the absolute value; the kernel metric will not be added. This option is used to make this router prefer HELLO routes learned via the specified interface(s) less than HELLO routes from other interfaces.

metricout metric

Specifies the HELLO metric to be added to routes that are sent via the specified interface(s). The default is zero. This option is used to make other routers prefer other sources of HELLO routes over this router.

trustedgateways gateway_list

Defines the list of gateways from which HELLO will accept updates. The gateway_list is simply a list of host names or IP addresses. By default, all routers on the shared network are trusted to supply routing information. But if the trustedgateways clause is specified, only updates from the gateways in the list are accepted.

sourcegateways gateway_list

Defines a list of routers to which HELLO sends packets directly, not through multicast or broadcast. This can be used to send different routing information to specific gateways. Updates to gateways in this list are not affected by noripout on the interface.

traceoptions trace_options

Specifies the tracing options for HELLO. (See Trace Statements and the HELLO specific tracing options below.)

The default preference is 90. The default metric is 30000.
Tracing options
The policy option logs info whenever a new route is announced, the metric being announced changes, or a route goes or leaves holddown.

Packet tracing options (which may be modified with detail, send, and/or recv):

packets All HELLO packets

The IS-IS Statement

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>isis no dual ip iso</td>
<td>Enables the IS-IS protocol in the gated daemon. By default, IS-IS is disabled. The dual option specifies that the IS-IS protocol is enabled for both ISO and IP addressing. The isis statement consists of an initial description of the IS and a list of statements that determine the configuration of the specific circuits and networks to be managed. Statements may appear in any order and include:</td>
</tr>
<tr>
<td>level 1</td>
<td>2</td>
</tr>
<tr>
<td>traceoptions</td>
<td>Covered in the Tracing options section below.</td>
</tr>
<tr>
<td>systemid</td>
<td>Overrides the autoconfigured system ID (determined from interface addresses and corresponding netmasks). If no system identifier is specified, the system ID portion of the first real circuit's NSAP is used. Once a system ID is set, it cannot be changed without disabling and reenabling all of IS-IS.</td>
</tr>
<tr>
<td>area</td>
<td>IS-IS area addresses are automatically configured based on the real circuits over which IS-IS runs. Addresses specified in this statement are maintained in addition to those configured automatically from the circuits. This command is used primarily for simulation.</td>
</tr>
<tr>
<td>circuit</td>
<td>Each circuit statement specifies one of the circuits the system will manage. Circuits normally correspond to UNIX interfaces, with string being the interface name, but simulated device names may also be specified. If the string is in the form of &quot;simN&quot;, where N is an integer, the circuit is assumed to be a simulated circuit managed by the network simulator troll. The circuit attributes are a list of options that may appear in any order in the circuit statement.</td>
</tr>
<tr>
<td>metric</td>
<td>Allows specifications of Level 1 and Level 2 metrics for each circuit. Only the default metric type is supported. IS-IS metrics must be in the range 1 to 63. If no metric is set for the circuit, the default value is 63.</td>
</tr>
<tr>
<td>priority</td>
<td>Determines designated router election results; higher values give a higher likelihood of becoming the designated router. The level defaults to Level 1. If no priority is specified, priority is set to a random value between 0 and 127.</td>
</tr>
</tbody>
</table>

On a level 2 IS, to configure a circuit with a Level 1 metric of 10 and a Level 2 metric of 20, add two metric options to the circuit statement.

The default Level is 1: the default metric is 63. The default preference for IS-IS Level 1 is 15 for IS-IS Level 2 is 18.
**Tracing options**

Traceoptions can be one or more of the following:

- all
- iih
- lanadj
- p2padj
- lspdb
- lspcontent
- lspinput
- flooding
- buildlsp
- csnp
- psnp
- route
- update
- paths
- spf
- events

**The OSPF Statement**

```bash
osfyes | no | on | off [{
  defaults {
    preference preference;
    cost cost;
    tag [as] tag;
    type 1 | 2;
  };
  exportlimit routes;
  exportinterval time;
  traceoptions trace_options;
  monitorauthkey authkey;
  monitorauth none | ( [simple | md5] authkey );
  backbone | ( area area ) { 
    authtype 0 | 1 | none | simple | md5;
    stub [cost cost];
    networks {
      network [restrict];
      network mask mask [restrict];
      network masklen number [restrict];
      host host [restrict];
    };
  stubhosts {
    host cost cost;
  };
  interface interface_list [cost cost] {
    interface_parameters
  };
  interface interface_list nonbroadcast [cost cost] {
    pollinterval time;
    routers {
      gateway [eligible];
    };
    interface_parameters
  };
  Backbone only:
  virtuallink neighborid router_id transitarea area {
    interface_parameters
  }
}
```
The following are the \textit{interface\_parameters} referred to above. They may be specified on any class of interface and are described under the \textit{interface} clause.

\begin{verbatim}
enable | disable;
retransmitinterval time;
transidelay time;
priority priority;
hellointerval time;
routerdelayinterval time;
authkey auth_key | auth md5 key auth_key id key_id ;
\end{verbatim}

\textbf{defaults}

These parameters specify the defaults used when importing OSPF ASE routes into the gated routing table and exporting routes from the gated routing table into OSPF ASEs.

\begin{itemize}
\item \textbf{preference} \texttt{preference}
  \textit{Preference} is used to determine how OSPF routes compete with routes from other protocols in the gated routing table. The default value is 150.
\item \textbf{cost} \texttt{cost}
  \textit{Cost} is used when exporting a non-OSPF route from the gated routing table into OSPF as an ASE. The default value is 1. This may be explicitly overridden in export policy.
\item \textbf{tag [as ]} \texttt{tag}
  OSPF ASE routes have a 32 bit tag field that is not used by the OSPF protocol, but may be used by export policy to filter routes. When OSPF is interacting with an EGP, the tag field may be used to propagate AS path information, in which case the \texttt{as} keyword is specified and the tag is limited to 12 bits of information. If not specified, the tag is set to zero.
\item \textbf{type 1 \& 2}
  Routes exported from the gated routing table into OSPF default to becoming type 1 ASEs. This default may be explicitly changed here and overridden in export policy.
\end{itemize}

\textbf{ASE export rate}

Because of the nature of OSPF, the rate at which ASEs are flooded must be limited. These two parameters can be used to adjust those rate limits.

\begin{itemize}
\item \textbf{exportinterval} \texttt{time}
  This specifies how often a batch of ASE link state advertisements will be generated and flooded into OSPF. The default is once per second.
\item \textbf{exportlimit} \texttt{routes}
  This parameter specifies how many ASEs will be generated and flooded in each batch. The default is 100.
\end{itemize}

\textbf{traceoptions} \texttt{trace\_options}

Specifies the tracing options for OSPF. (See \texttt{Trace Statements} and the OSPF specific tracing options below.)

\textbf{monitorauthkey} \texttt{authkey}

OSPF state may be queried using the \texttt{ospf\_monitor} command utility. This utility sends non-standard OSPF packets that generate a text response from OSPF. By default, these requests are not authenticated if an authentication key is configured, the incoming requests must match the specified authentication key. No OSPF state may be changed by these packets, but the act of querying OSPF can utilize system resources.

\textbf{backbonearea} \texttt{area}

Each OSPF router must be configured into at least one OSPF area. If more than one area is
configured, at least one must be backbone. The backbone may only be configured using the `backbone` keyword, it may not be specified as area 0. The backbone interface may be a virtual link.

**authype** 0 | 1 | none | simple | md5

OSPF specifies an authentication scheme per area. Each interface in the area must use this same authentication scheme although it may use a different authentication key. The currently valid values are none (0) for no authentication, simple (1) for simple password authentication or md5 for MD5 authentication.

**stub [cost cost]**

A stub area is one in which there are no ASE routes. If a cost is specified, this is used to inject a default route into the area with the specified cost.

**networks**

The `networks` list describes the scope of an area. Intra-area LSAs that fall within the specified ranges are not advertised into other areas as inter-area routes. Instead, the specified ranges are advertised as summary network LSAs. If `restrict` is specified, the summary network LSAs are not advertised. Intra-area LSAs that do not fall into any range are also advertised as summary network LSAs. This option is very useful on well designed networks in reducing the amount of routing information propagated between areas. The entries in this list are either networks, or a subnetwork/mask pair. See the section on [Route Filtering](#) for more detail about specifying ranges.

**stubhosts**

This list specifies directly attached hosts that should be advertised as reachable from this router and the costs they should be advertised with. Point-to-point interfaces on which it is not desirable to run OSPF should be specified here.

It is also useful to assign an additional address to the loopback interface (one not on the 127 network) and advertise it as a stub hosts. If this address is the same one used as the router-id, it enables routing to OSPF routers by router-id, instead of by an interface address. This is more reliable than routing to one of the router’s interface addresses that may not always be reachable.

**interface** `interface_list` [cost cost]

This form of the interface clause is used to configure a broadcast (which requires IP multicast support) or a point-to-point interface. See the section on interface list specification for the description of the `interface_list`.

Each interface has a cost. The costs of all interfaces a packet must cross to reach a destination are summed to get the cost to that destination. The default cost is one, but another non-zero value may be specified.

Interface parameters common to all types of interfaces are:

**retransmitinterval** `time`

The number of seconds between link state advertisement retransmissions for adjacencies belonging to this interface.

**transitdelay** `time`

The estimated number of seconds required to transmit a link state update over this interface. Transitdelay takes into account transmission and propagation delays and must be greater than 0.

**priority** `priority`

A number between 0 and 255 specifying the priority for becoming the designated router on this interface. When two routers attached to a network both attempt to become the designated router, the one with the highest priority wins. A router whose router priority is set to 0 is ineligible to become the designated router.
**helloat**

The length of time, in seconds, between Hello packets that the router sends on the interface.

**routerdeadinterval**

The number of seconds not hearing a router’s Hello packets before the router’s neighbors will declare it down.

**authkey auth_key I auth md5 key auth_key id key_id ;**

Used by OSPF authentication to generate and verify the authentication field in the OSPF header. The authentication key can be configured on a per interface basis. It is specified by one to eight decimal digits separated by periods, a one to eight byte hexadecimal string preceded by 0x, or a one to eight character string in double quotes.

For MD5 authentication, the auth_key is specified by a 1 to 8 character string in double quotes. The id specifies the algorithm used by MD5 to calculate the message-digest and its value ranges from 1 to 255.

Point-to-point interfaces also support this additional parameter:

**nomulticast**

By default, OSPF packets to neighbors on point-to-point interfaces are sent via the IP multicast mechanism. If the use of IP multicasting is not desired, the nomulticast parameter may be specified to force the use of unicast OSPF packets. gated.conf will detect this condition and fall back to using sending unicast OSPF packets to this point-to-point neighbor.

If the use of IP multicasting is not desired because the remote neighbor does not support it, the nomulticast parameter may be specified to force the use of unicast OSPF packets. This option may also be used to eliminate warnings when gated.conf detects the bug mentioned above.

**interface interface_list nonbroadcast [cost cost ]**

This form of the interface clause is used to specify a nonbroadcast interface on a non-broadcast multi-access (NBMA) media. Since an OSPF broadcast media must support IP multicasting, a broadcast-capable media, such as Ethernet, that does not support IP multicasting must be configured as a non-broadcast interface.

A non-broadcast interface supports any of the standard interface clauses listed above, plus the following two that are specific to non-broadcast interfaces:

**pollinterval time**

Before adjacency is established with a neighbor, OSPF packets are sent periodically at the specified pollinterval.

**routers**

By definition, it is not possible to send broadcast packets to discover OSPF neighbors on a non-broadcast, so all neighbors must be configured. The list includes one or more neighbors and an indication of their eligibility to become a designated router.

**virtuallink neighborid router_id transitarea area**

Virtual links are used to establish or increase connectivity of the backbone area. The neighborid is the router_id of the other end of the virtual link. The transit area specified must also be configured on this system. All standard interface parameters defined by the interface clause above may be specified on a virtual link.

**Tracing options**

In addition to the following OSPF specific trace flags, OSPF supports the state that traces interface and neighbor state machine transitions.

**Isabuild**

Link State Advertisement creation
Packet tracing options (which may be modified with `detail`, `send` and `recv`):

- **hello**: OSPF HELLO packets that are used to determine neighbor reachability.
- **dd**: OSPF Database Description packets that are used in synchronizing OSPF databases.
- **request**: OSPF Link State Request packets that are used in synchronizing OSPF databases.
- **lsu**: OSPF Link State Update packets that are used in synchronizing OSPF databases.
- **ack**: OSPF Link State Ack packets that are used in synchronizing OSPF databases.

### The EGP Statement

```
EGP yes | no | on | off
{}
  preference preference ;
  defaultmetric metric ;
  packetsize number ;
  traceoptions trace_options ;
  group
    [peeras autonomous_system ]
    [localas autonomous_system ]
    [maxup number ]
  {neighbor host
    [metricout metric ]
    [preference preference ]
    [preference2 preference ]
    [ttl ttl ]
    [nogendefault ]
    [importdefault ]
    [exportdefault ]
    [gateway gateway ]
    [lcladdr local_address ]
    [sourcenet network ]
    [minhello | p1 time ]
    [minpoll | p2 time ]
    [traceoptions trace_options ]
  ;
  }
}
```

**preference** `preference`

Sets the preference for routes learned from RIP. The default preference is 200. This preference may be overridden by a preference specified on the `group` or `neighbor` statements or by import policy.

**defaultmetric** `metric`;

Defines the metric used when advertising routes via EGP. If not specified, the default value is 255 that some systems may consider unreachable. This choice of values requires you to explicitly specify a metric when exporting routes to EGP neighbors. This metric may be overridden by a metric specified on the neighbor or group statements or in export policy.

**packetsize** `maxpacketsize`

This defines the expected maximum size of a packet that EGP expects to receive from this neighbor. If a packet larger than this value is received, it will be incomplete and have to be discarded. The length of this packet will be noted and the expected size will be increased to be

---

**spf**

Shortest Path First (SPF) calculations
able to receive a packet of this size. Specifying the parameter here will prevent the first packet from being dropped. If not specified, the default size is 8192 bytes. All packet sizes are rounded up to a multiple of the system page size.

**traceoptions**

Specifies the tracing options for EGP. By default these are inherited from the global trace options. These values may be overridden on a group or neighbor basis. (See [Trace Statements](#) and the EGP specific tracing options below.)

**group**

EGP neighbors must be specified as members of a `group`. A group is usually used to group all neighbors in one autonomous system. Parameters specified on the group clause apply to all of the subsidiary neighbors unless explicitly overridden on a neighbor clause. Any number of `group` clauses may specify any number of `neighbor` clauses.

Any parameters from the `neighbor` subclause may be specified on the `group` clause to provide defaults for the whole group (which may be overridden for individual neighbors). In addition, the `group` clause is the only place to set the following attributes:

**peeras**

Identifies the autonomous system number expected from peers in the group. If not specified, it will be learned dynamically.

**localas**

Identifies the autonomous system that `gated.conf` is representing to the group. The default is that which has been set globally in the `autonomous-system` statement. This option is usually only used when *masquerading* as another autonomous system and its use is discouraged.

**maxup**

Specifies the number of neighbors the `gated` daemon should acquire from this group. The default is to acquire all of the neighbors in the group. The `gated` daemon will attempt to acquire the first `maxup` neighbors in the order listed. If one of the first neighbors is not available, it will acquire one further down the list. If after start-up the `gated` daemon does manage to acquire the more desirable neighbor, it will drop the less desirable one.

**neighbor**

Each neighbor subclause defines one EGP neighbor within a group. The only part of the subclause that is required is the `neighbor_address` argument that is the symbolic host name or IP address of the neighbor. All other parameters are optional.

**preference**

Specifies the preference used for routes learned from these neighbors. This can differ from the default EGP preference set in the `EGP` statement, so that the `gated` daemon can prefer routes from one neighbor, or group of neighbors, over another. This preference may be explicitly overridden by import policy.

**preference2**

In the case of a preference tie, the second preference, `preference2` may be used to break the tie. The default value is 0.

**metricout**

This defines a metric to be used for all routes sent to this neighbor. The value overrides the default metric set in the `EGP` statement and any metrics specified by export policy, but only for this specific neighbor or group of neighbors.

**nogendefault**

Prevents `gated.conf` from generating a default route when EGP receives a valid update from its neighbor. The default route is only generated when the `gendefault` option is enabled.

**importdefault**

Enables the `gated` daemon to accept the default route (0.0.0.0) if it is included in a
received EGP update. If not specified, the default route contained in an EGP update is ignored. For efficiency, some networks have external routers announce a default route to avoid sending large EGP update packets.

**exportdefault**
Enables the gated daemon to include the default route (0.0.0.0) in EGP updates sent to this EGP neighbor. This allows the system to advertise the default route via EGP. Normally a default route is not included in EGP updates.

**gateway gateway**
If a network is not shared with a neighbor, gateway specifies a router on an attached network to be used as the next hop router for routes received from this neighbor. This option is only rarely used.

**lcladdr local_address**
Specifies the address to be used on the local end of the connection with the neighbor. The local address must be on an interface that is shared with the neighbor or with the neighbor's gateway when the gateway parameter is used. A session will only be opened when an interface with the appropriate local address (through which the neighbor or gateway address is directly reachable) is operating.

**sourcenet network**
Specifies the network queried in the EGP Poll packets. By default, this is the network shared with the neighbor's address specified. If there is no network shared with the neighbor, one of the networks the neighbor is attached to should be specified. This parameter can also be used to specify a network shared with the neighbor other than the one on which the EGP packets are sent. This parameter is normally not needed.

**p1 time**

**minhello time**
Sets the minimum acceptable interval between the transmission of EGP HELLO packets. The default hello interval is 30 seconds. If the neighbor fails to respond to three hello packets, the gated daemon stops trying to acquire the neighbor. Setting a larger interval gives the neighbor a better chance to respond. Minhello is an alias for the P1 value defined in the EGP specification.

**p2 time**

**minpoll time**
Sets the time interval between polls to the neighbor. The default is 120 seconds. If three polls are sent without a response, the neighbor is declared "down" and all routes learned from that neighbor are removed from the routing database. A longer polling interval supports a more stable routing database but is not as responsive to routing changes. Minpoll is an alias for the P2 value defined in the EGP specification.

**ttl ttl**
By default, the gated daemon sets the IP TTL for local neighbors to one and the TTL for non-local neighbors to 255. This option is provided when attempting to communicate with improperly functioning routers that ignore packets sent with a TTL of one.

**traceoptions trace_options**
Specifies the tracing options for this EGP neighbor. By default, these are inherited from group or EGP global trace options. (See Trace Statements and the EGP specific tracing options below.)

**Tracing options**
The state and policy options work with EGP.

Packet tracing options (which may be modified with detail, send and recv):

**packets**
All EGP packets
The BGP Statement

```
bgp  yes | no | on | off
[
    preference preference ;
    defaultmetric metric ;
    traceoptions trace_options ;
    group  type  (
        external peers autonomous_system )
    | ( internal peers autonomous_system )
    | ( IGP peers autonomous_system proto proto )
    | ( routing peers autonomous_system proto peer
        interface interface_list )
    | ( test peers autonomous_system )
    { allow 
        network
        network mask mask
        network masklen number
        all
        host host
    } ;
    peer  host
    [ metricout metric ]
    [ locals autonomous_system ]
    [ nogendefault]
    [ gateway gateway ]
    [ preference preference ]
    [ preference2 preference ]
    [ lcladdr local_address ]
    [ holdtime time ]
    [ version number ]
    [ passive]
    [ indelay time ]
    [ outdelay time ]
    [ keep [ all | none] ]
    [ noaggregatorid]
    [ keepalivesalways]
    [ v3asloopokay]
    [ nov4asloop]
    [ logupdown]
    [ ttl ttl ]
    [ traceoptions trace_options ]
    ;
}
```

The `bgp` statement enables or disables BGP. By default, BGP is disabled. The default metric for announcing routes via BGP is not to send a metric.
Sets the preference for routes learned from RIP. The default preference is 170. This preference may be overridden by a preference specified on the group or peer statements or by import policy.

`defaultmetric metric` Defines the metric used when advertising routes via BGP. If not specified, no metric is propagated. This metric may be overridden by a metric specified on the neighbor or group statements or in export policy.

`traceoptions trace_options` Specifies the tracing options for BGP. By default these are inherited from the global trace options. These values may be overridden on a group or neighbor basis. (See Trace Statements and the BGP specific tracing options below.)

Groups
BGP peers are grouped by type and the autonomous system of the peers. Any number of groups may be specified, but each must have a unique combination of type and peer autonomous system. There are four possible group types:

**group type external peeras autonomous_system**
In the classic external BGP group, full policy checking is applied to all incoming and outgoing advertisements. The external neighbors must be directly reachable through one of the machine’s local interfaces. By default no metric is included in external advertisements, and the next hop is computed with respect to the shared interface.

**group type internal peeras autonomous_system**
An internal group operating where there is no IP-level IGP. All neighbors in this group are required to be directly reachable via a single interface. All next hop information is computed with respect to this interface. Import and export policy may be applied to group advertisements. Routes received from external BGP or EGP neighbors are by default readvertised with the received metric.

**group type IGPeer peeras autonomous_system proto proto**
An internal group that runs in association with an interior protocol. The IGP group examines routes that the IGP is exporting and sends an advertisement only if the path attributes could not be entirely represented in the IGP tag mechanism. Only the AS path, path origin, and transitive optional attributes are sent with routes. No metric is sent, and the next hop is set to the local address used by the connection. Received internal BGP routes are not used or readvertised. Instead, the AS path information is attached to the corresponding IGP route and the latter is used for readvertisement. Since internal IGP peers are sent only a subset of the routes that the IGP is exporting, the export policy used is the IGP’s. There is no need to implement the “don’t routes from peers in the same group” constraint since the advertised routes are routes that IGP already exports.

**group type routing peeras autonomous_system proto proto interface interface_list**
An internal group that uses the routes of an interior protocol to resolve forwarding addresses. A type routing group propagates external routes between routers that are not directly connected, and computes immediate next hops for these routes by using the BGP next hop that arrived with the route as a forwarding address to be resolved via an internal protocol's routing information. In essence, internal BGP is used to carry AS external routes, while the IGP is expected to only carry AS internal routes, and the latter is used to find immediate next hops for the former.

The `proto` names the interior protocol to be used to resolve BGP route next hops, and may be the name of any IGP in the configuration. By default, the next hop in BGP routes advertised to type routing peers will be set to the local address on the BGP connection to those peers, as it is assumed a route to this address will be propagated via the IGP. The `interface_list` can optionally provide list interfaces whose routes are carried via the IGP for which third party next hops may be used instead.

**group type test peeras autonomous_system**
An extension to external BGP that implements a fixed policy using test peers. Fixed policy and
special case code make test peers relatively inexpensive to maintain. Test peers do not need to be on a directly attached network. If the gated daemon and the peer are on the same (directly attached) subnet, the advertised next hop is computed with respect to that network, otherwise the next hop is the local machine’s current next hop. All routing information advertised by and received from a test peer is discarded, and all BGP advertisable routes are sent back to the test peer. Metrics from EGP- and BGP-derived routes are forwarded in the advertisement, otherwise no metric is included.

Group parameters
The BGP statement has group clauses and peer subclauses. Any number of peer subclauses may be specified within a group. A group clause usually defines default parameters for a group of peers, these parameters apply to all subsidiary peer subclauses. Any parameters from the peer subclause may be specified on the group clause to provide defaults for the whole group (which may be overridden for individual peers).

Specifying peers
Within a group, BGP peers may be configured in one of two ways. They may be explicitly configured with a peer statement, or implicitly configured with the allow statement. Both are described here:

allow The allow clause allows for peer connections from any addresses in the specified range of network and mask pairs. All parameters for these peers must be configured on the group clause. The internal peer structures are created when an incoming open request is received and destroyed when the connection is broken. For more detail on specifying the network/mask pairs, see the section on Route Filtering.

peer host A peer clause configures an individual peer. Each peer inherits all parameters specified on a group as defaults. Those defaults may be overridden by parameters explicitly specified on the peer subclause.

Within each group clause, individual peers can be specified or a group of potential peers can be specified using allow. Allow is used to specify a set of address masks. If the gated daemon receives a BGP connection request from any address in the set specified, it will accept it and set up a peer relationship.

Peer parameters
The BGP peer subclause allows the following parameters, which can also be specified on the group clause. All are optional.

metricout metric
If specified, this metric is used as the primary metric on all routes sent to the specified peer(s). This metric overrides the default metric, a metric specified on the group and any metric specified by export policy.

localas autonomous_system
Identifies the autonomous system that the gated daemon is representing to this group of peers. The default is that which has been set globally in the autonomous_system statement.

nogendefault
Prevents gated.conf from generating a default route when EGP receives a valid update from its neighbor. The default route is only generated when the gendefault option is enabled.

gateway gateway
If a network is not shared with a peer, gateway specifies a router on an attached network to be used as the next hop router for routes received from this neighbor. This parameter is not needed in most cases.

preference preference
Specifies the preference used for routes learned from these peers. This can differ from the default
BGP preference set in the bgp statement, so that the gated daemon can prefer routes from one peer, or group of peer, over others. This preference may be explicitly overridden by import policy.

**preference2 preference**
In the case of a preference tie, the second preference, preference2 may be used to break the tie. The default value is 0.

**lcladdr local_address**
Specifies the address to be used on the local end of the TCP connection with the peer. For external peers the local address must be on an interface that is shared with the peer or with the peer's gateway when the gateway parameter is used. A session with an external peer will only be opened when an interface with the appropriate local address (through which the peer or gateway address is directly reachable) is operating. For other types of peers, a peer session will be maintained when any interface with the specified local address is operating. In either case, incoming connections will only be recognized as matching a configured peer if they are addressed to the configured local address.

**holdtime time**
Specifies the BGP holdtime value to use when negotiating the connection with this peer, in seconds. According to BGP, if the gated daemon does not receive a keepalive, update, or notification message within the period specified in the Hold Time field of the BGP Open message, then the BGP connection will be closed. The value must be either 0 (no keepalives will be sent) or at least 3.

**version version**
Specifies the version of the BGP protocol to use with this peer. If not specified, the highest supported version is used first and version negotiation is attempted. If it is specified, only the specified version will be offered during negotiation. Currently supported versions are 2, 3 and 4.

**passive**
Specifies that active OPENs to this peer should not be attempted. the gated daemon should wait for the peer to issue an OPEN. By default, all explicitly configured peers are active, they periodically send OPEN messages until the peer responds.

**indelay time**
**outdelay time**
Used to dampen route fluctuations. Indelay is the amount of time a route learned from a BGP peer must be stable before it is accepted into the gated routing database. Outdelay is the amount of time a route must be present in the gated routing database before it is exported to BGP. The default value for each is 0, meaning that these features are disabled.

**keep all**
Used to retain routes learned from a peer even if the routes' AS paths contain one of our exported AS numbers.

**noaggregatorid**
Causes the gated daemon to specify the routerid in the aggregator attribute as zero (instead of its routerid) in order to prevent different routers in an AS from creating aggregate routes with different AS paths.

**keepalivesalways**
Causes the gated daemon to always send keepalives, even when an update could have correctly substituted for one. This allows interoperability with routers that do not completely obey the protocol specifications on this point.

**v3asloopokay**
By default, the gated daemon will not advertise routes whose AS path is looped (that is, with an AS appearing more than once in the path) to version 3 external peers. Setting this flag removes this constraint. Ignored when set on internal groups or peers.
nov4asloop
Prevents routes with looped AS paths from being advertised to version 4 external peers. This can be useful to avoid advertising such routes to peers that would incorrectly forward the routes on to version 3 neighbors.

logupdown
Causes a message to be logged via the syslog mechanism whenever a BGP peer enters or leaves the ESTABLISHED state.

traceoptions trace_options
Specifies the tracing options for this BGP neighbor. By default, these are inherited from group or BGP global trace options. (See Trace Statements and the BGP specific tracing options below.)

Tracing options
Note: The state option works with BGP, but does not provide true state transition information.

Packet tracing options (which may be modified with detail, send, and recv):

- packets: All BGP packets.
- open: BGP OPEN packets that are used to establish a peer relationship.
- update: BGP UPDATE packets that are used to pass network reachability information.
- keepalive: BGP KEEPALIVE packets that are used to verify peer reachability.

The BGP4+ Statement
The options are the same for BGP4+ as they are for bgp but all the addresses will be IPv6 addresses.

The syntax is:

```
bgp4+ yes | no | on | off
[
  preference <preference> ;
  defaultmetric <metric> ;
  traceoptions <trace_options> ;
  group type ( external peers <autonomous_system> )
    ( internal peers <autonomous_system> )
    ( igp peers <autonomous_system> proto <proto> )
    ( routing peers <autonomous_system> proto <proto>
      interface <interface_list> )
    ( test peers <autonomous_system> )
  ]
  allow {
    <network>
    masklen <number>
    all
    host <IPv6 host address>
  };
  peer <IPv6 host address>
    [ metricout <metric> ]
    [ localas <autonomous_system> ]
    [ nogendefault ]
    [ gateway <gateway> ]
    [ preference <preference> ]
    [ preference2 <preference> ]
    [ lcladdr <local_address> ]
    [ holdtime <time> ]
    [ version <number> ]
    [ passive ]
    [ sendbuffer <number> ]
    [ recvbuffer <number> ]
    [ indelay <time> ]
```
The ICMP Statement

```
icmp { 
  traceoptions trace_options; 
}
```

Specifies the tracing options for ICMP. (See Trace Statements and the ICMP specific tracing options below.)

**Tracing options**

Packet tracing options (which may be modified with `detail` and `recv`):

- **packets**
  
  All ICMP packets received.

- **redirect**
  
  Only ICMP `REDIRECT` packets received.

- **routerdiscovery**
  
  Only ICMP `ROUTER DISCOVERY` packets received.

- **info**
  
  Only ICMP informational packets, which include mask request/response, info request/response, echo request/response, and time stamp request/response.

- **error**
  
  Only ICMP error packets, which include time exceeded, parameter problem, unreachable and source quench.

The SNMP Statement

The Simple Network Management Protocol (SNMP) is not a routing protocol but a network management protocol. The `snmp` statement controls whether `gated.conf` tries to contact the SNMP Multiplexing daemon to register supported variables. The SNMP daemon, `smuxd`, must be run independently. The `snmp` statement only controls whether `gated.conf` keeps the management software apprised of its status.

```
gated.conf communicates with the SNMP daemon via the SMUX protocol that is described in RFC 1227.
gated.conf yes | no | on | off
[
  port port ;
  debug;
  traceoptions trace_options;
] ;
```

Reporting is enabled by specifying yes or on and disabled with no or off. The default is on.

- **port port**
  
  By default, the SMUX daemon listens for requests on port 199. The `gated.conf` subroutine can be configured to try to contact the SMUX daemon on a different port by explicitly specifying the port. Specifying this option enables debugging of the ISODE SMUX code. The default is debugging disabled.

- **traceoptions trace_options**
  
  Specifies the tracing options for SMUX. (See Trace Statements and the SMUX specific tracing options below.)
Tracing options
There are no SNMP-specific trace options. The detail, send, and recv options are not supported.

receive  SNMP requests received from the SMUX daemon and the associated responses.
register  Protocol requests to register variables.
resolve  Protocol requests to resolve variable names.
trap     SNMP trap requests from protocols.

Static Statements
Static statements define the static routes used by the gated daemon. A single static statement can specify any number routes. The static statements occur after protocol statements and before control statements in the gated.conf file. Any number of static statements may be specified, each containing any number of static route definitions. These routes can be overridden by routes with better preference values.

```
static {
    ( host host ) | default |
    ( network [ ( mask mask ) | ( masklen number ) ]
        gateway gateway_list
            [ interface interface_list ]
            [ preference preference ]
            [ retain]
            [ reject]
        [ blackhole]
            [ noinstall];
    ( network [ ( mask mask ) | ( masklen number ) ]
        interface interface
            [ preference preference ]
            [ retain]
            [ reject]
        [ blackhole]
            [ noinstall];
} ;
host host gateway gateway_list ( network [ ( mask mask ) | ( masklen number ) ] ) default gateway

This is the most general form of the static statement. It defines a static route through one or more gateways. Static routes are installed when one or more of the gateways listed are available on directly attached interfaces.

Parameters for static routes are:

interface [interface_list]
When this parameter is specified, gateways are only considered valid when they are on one of these interfaces. See the section on interface_list specification for the description of the interface_list.

preference [preference]
This option selects the preference of this static route. The preference controls how this route competes with routes from other protocols. The default preference is 60.

retain
Normally the gated daemon removes all routes except interface routes from the kernel forwarding table during a graceful shutdown. The retain option may be used to prevent specific static routes from being removed. This is useful to insure that some routing is available when gated is not running.
reject

Instead of forwarding a packet like a normal route, reject routes cause packets to be dropped and unreachable messages to be sent to the packet originators. Specifying this option causes this route to be installed as a reject route. Not all kernel forwarding engines support reject routes.

blackhole

A blackhole route is the same as a reject route except that unreachable messages are not supported.

noinstall

Normally the route with the lowest preference is installed in the kernel forwarding table and is the route exported to other protocols. When noinstall is specified on a route, it will not be installed in the kernel forwarding table when it is active, but it will still be eligible to be exported to other protocols.

(network [mask mask] | (masklen number)) interface interface

This form defines a static interface route that is used for primitive support of multiple network addresses on one interface. The preference, retain, reject, blackhole and noinstall options are the same as described above.

The Import Statement

Importation of routes from routing protocols and installation of the routes in the gated daemon’s routing database is controlled by import statements. The format of an import statement varies depending on the source protocol.

Specifying preferences

In all cases, one of two keywords may be specified to control how routes compete with other protocols:

restrict preference preference

restrict

Specifies that the routes are not desired in the routing table. In some cases, this means that the routes are not installed in the routing table. In others, it means that they are installed with a negative preference; this prevents them from becoming active so they will not be installed in the forwarding table, or exported to other protocols.

preference preference

Specifies the preference value used when comparing this route to other routes from other protocols. The route with the lowest preference available at any given route becomes the active route, is installed in the forwarding table, and is eligible to be exported to other protocols. The default preferences are configured by the individual protocols.

Route Filters

All the formats allow route filters as shown below. See the section on route filters for a detailed explanation of how they work. When no route filtering is specified (that is, when restrict is specified on the first line of a statement), all routes from the specified source will match that statement. If any filters are specified, only routes that match the specified filters will be imported. Put differently, if any filters are specified, an all restrict; is assumed at the end of the list.

network [exact | refines]
network mask mask [exact | refines]
network masklen number [exact | refines]
default
host host

Importing Routes from BGP and EGP

import proto bgp | EGP autonomoussystem autonomous_system
restrict;
import proto bgp | EGP autonomousystem autonomous_system
  [ preference preference ]{
    route_filter [ restrict | ( preference preference ) ];
  }

import proto bgp aspath aspath_regexp
  origin any | ( [ IGP ] [ EGP ] [ incomplete ] )
  restrict;
import proto bgp aspath aspath_regexp
  origin any | ( [ IGP ] [ EGP ] [ incomplete ] )
  [ preference preference ]{
    route_filter [ restrict | ( preference preference ) ];
  }
EGP importation may be controlled by autonomous system.

BGP also supports controlling propagation by the use of AS path regular expressions, which are
documented in the section on [Matching AS paths]

Note: EGP and BGP versions 2 and 3 only support the propagation of natural networks, so the host and
default route filters are meaningless. BGP version 4 supports the propagation of any destination
along with a contiguous network mask.

EGP and BGP both store any routes that were rejected implicitly by not being mentioned in a route filter,
or explicitly with the restrict keyword in the routing table with a negative preference. A negative
preference prevents a route from becoming active, which prevents it from being installed in the forwarding
table, or exported to other protocols. This alleviates the need to break and re-establish a session upon
reconfiguration if importation policy is changed.

Importing Routes from RIP, HELLO and Redirects
import proto rip | hello | redirect
  [ ( interface interface_list ) | (gateway gateway_list ) ]
  restrict;
import proto rip | hello | redirect
  [ ( interface interface_list ) | (gateway gateway_list ) ]
  [ preference preference ]{
    route_filter [ restrict | ( preference preference ) ];
  }

The importation of RIP, HELLO, and Redirect routes may be controlled by any of protocol, source
interface, and source gateway. If more than one is specified, they are processed from most general
(protocol) to most specific (gateway).

RIP and HELLO don’t support the use of preference to choose between routes of the same protocol. That
is left to the protocol metrics. These protocols do not save routes that were rejected since they have short
update intervals.

Importing Routes from OSPF
import proto ospfase [ tag ospf_tag ] restrict;
import proto ospfase [ tag ospf_tag ]
  [ preference preference ]{
    route_filter [ restrict | ( preference preference ) ];
  }

Due to the nature of OSPF, only the importation of ASE routes may be controlled. OSPF intra- and inter-area routes are always imported into the gated routing table with a preference of 10. If a tag is specified, the import clause will only apply to routes with the specified tag.

It is only possible to restrict the importation of OSPF ASE routes when functioning as an AS border router. This is accomplished by specifying an export ospfase clause. Specification of an empty export clause may be used to restrict importation of ASEs when no ASEs are being exported.

Like the other interior protocols, preference can not be used to choose between OSPF ASE routes, that is done by the OSPF costs. Routes that are rejected by policy are stored in the table with a negative preference.

The Export Statement

The import statement controls routes received from other systems that are used by the gated daemon, and the export statement controls which routes are advertised by the gated daemon to other systems. Like the import statement, the syntax of the export statement varies slightly per protocol. The syntax of the export statement is similar to the syntax of the import statement, and the meanings of many of the parameters are identical. The main difference between the two is that while route importation is just controlled by source information, route exportation is controlled by both destination and source.

The outer portion of a given export statement specifies the destination of the routing information you are controlling. The middle portion restricts the sources of importation that you wish to consider And the innermost portion is a route filter used to select individual routes.

Specifying Metrics

The most specific specification of a metric is the one applied to the route being exported. The values that may be specified for a metric depend on the destination protocol that is referenced by this export statement.

```
restrict
metric metric
```

restrict Specifies that nothing should be exported. If specified on the destination portion of the export statement, it specifies that nothing at all should be exported to this destination. If specified on the source portion, it specifies that nothing from this source should be exported to this destination. If specified as part of a route filter, it specifies that the routes matching that filter should not be exported.

```
metric metric
```

Specifies the metric to be used when exporting to the specified destination.

Route Filters

All the formats allow route filters as shown below. See the section on route filters for a detailed explanation of how they work. When no route filtering is specified (that is, when restrict is specified on the first line of a statement), all routes from the specified source will match that statement. If any filters are specified, only routes that match the specified filters will be exported. Put differently, if any filters are specified, an all restrict ; is assumed at the end of the list.

```
network [ exact | refines ]
network mask mask [ exact | refines ]
network masklen number [ exact | refines ]
default
default
host host
```
Specifying the Destination

As mentioned above, the syntax of the export statement varies depending on the protocol to which it is being applied. One thing that applies in all cases is the specification of a metric. All protocols define a default metric to be used for routes being exported, in most cases this can be overridden at several levels of the export statement.

The specification of the source of the routing information being exported (the export_list) is described below.

Exporting to EGP and BGP

```
export proto bgp | EGP as autonomous system
    restrict;
export proto bgp | EGP as autonomous system
    [ metric metric ] {
        export_list;
    };
```

Exportation to EGP and BGP is controlled by autonomous system, the same policy is applied to all routers in the AS.

EGP metrics range from 0 to 255 inclusive with 0 being the most attractive.

BGP metrics are 16 bit unsigned quantities, that is, they range from 0 to 65535 inclusive with 0 being the most attractive.

If no export policy is specified, only routes to attached interfaces will be exported. If any policy is specified, the defaults are overridden. It is necessary to explicitly specify everything that should be exported.

**Note:** EGP and BGP versions 2 and 3 only support the propagation of natural networks, so the host and default route filters are meaningless. BGP version 4 supports the propagation of any destination along with a contiguous network mask.

Exporting to RIP and HELLO

```
export proto rip | hello
    [ ( interface interface_list ) | (gateway gateway_list ) ]
    restrict;
export proto rip | hello
    [ ( interface interface_list ) | (gateway gateway_list ) ]
    [ metric metric ] {
        export_list;
    };
```

Exportation to RIP and HELLO is controlled by any of protocol, interface or gateway. If more than one is specified, they are processed from the most general (protocol) to the most specific (gateway).

It is not possible to set metrics for exporting RIP routes into RIP, or exporting HELLO routes into HELLO. Attempts to do this are silently ignored.

If no export policy is specified, RIP and interface routes are exported into RIP and HELLO and interface routes are exported into HELLO. If any policy is specified, the defaults are overridden. It is necessary to explicitly specify everything that should be exports.

RIP version 1 and HELLO assume that all subnets of the shared network have the same subnet mask so they are only able to propagate subnets of that network. RIP version 2 removes that restriction and is capable of propagating all routes when not sending version 1 compatible updates.
To announce routes that specify a next hop of the loopback interface (that is, static and internally generated default routes) via RIP or HELLO, it is necessary to specify the metric at some level in the export clause. For example, just setting a default metric for RIP or HELLO is not sufficient. This is a safeguard to verify that the announcement is intended.

Exporting to OSPF

```
export proto osfpase [ type 1 | 2 ] [ tag ospf_tag ]
    restrict ;
export proto osfpase [ type 1 | 2 ] [ tag ospf_tag ]
    [ metric metric ] { export_list ; }
```

It is not possible to create OSPF intra- or inter-area routes by exporting routes from the the gated daemon routing table into OSPF. It is only possible to export from the gated daemon routing table into OSPF ASE routes. It is also not possible to control the propagation of OSPF routes within the OSPF protocol.

There are two types of OSPF ASE routes, type 1 and type 2. See the OSPF protocol configuration for a detailed explanation of the two types. The default type is specified by the defaults subclause of the ospf clause. This may be overridden by a specification on the export statement.

OSPF ASE routes also have the provision to carry a tag. This is an arbitrary 32 bit number that can be used on OSPF routers to filter routing information. See the OSPF protocol configuration for detailed information on OSPF tags. The default tag specified by the ospf defaults clause may be overridden by a tag specified on the export statement.

Specifying the Source

The export list specifies export based on the origin of a route and the syntax varies depending on the source.

Exporting BGP and EGP Routes

```
proto bgp | EGP autonomous_system autonomous_system
    restrict ;
proto bgp | EGP autonomous_system autonomous_system
    [ metric metric ] { route_filter [ restrict | ( metric metric ) ] ; }
```

BGP and EGP routes may be specified by the source autonomous system. All routes may be exported by as path, see the Exporting by AS Path section for more information.

Exporting RIP and HELLO Routes

```
proto rip | hello
    [ ( interface interface_list ) | (gateway gateway_list ) ]
    restrict ;
proto rip | hello
    [ ( interface interface_list ) | (gateway gateway_list ) ]
    [ metric metric ] { route_filter [ restrict | ( metric metric ) ] ; }
```

RIP and HELLO routes may be exported by protocol, source interface, and/or source gateway.
Exporting OSPF Routes

```
proto ospf | ospfase restrict ;
proto ospf | ospfase [ metric metric ] {
    route_filter [ restrict | ( metric metric ) ] ;
}
```

Both OSPF and OSPF ASE routes may be exported into other protocols. See below for information on exporting by tag.

Exporting Routes from Non-routing Protocols

Non-routing with Interface

```
proto direct | static | kernel
    [ ( interface interface_list ) ]
    restrict ;
proto direct | static | kernel
    [ ( interface interface_list ) ]
    [ metric metric ] {
        route_filter [ restrict | ( metric metric ) ] ;
    }
```

These protocols may be exported by protocol, or by the interface of the next hop. These protocols are:

- **direct**: Routes to directly attached interfaces.
- **static**: Static routes specified in a `static` clause.
- **kernel**: Routes learned from the routing socket are installed in the gated routing table with a protocol of `kernel`. These routes may be exported by referencing this protocol.

Non-routing by Protocol

```
proto default | aggregate
    restrict ;
proto default | aggregate
    [ metric metric ] {
        route_filter [ restrict | ( metric metric ) ] ;
    }
```

These protocols may only be referenced by protocol.

- **default**: Refers to routes created by the `gendefault` option. It is recommended that route generation be used instead.
- **aggregate**: Refers to routes synthesized from other routes when the `aggregate` and `generate` statements are used. See the section on Route Aggregation for more information.

Exporting by AS Path

```
proto proto | all aspath aspath_regexp
    origin any | ( ( [ IGP ] [ EGP ] [ incomplete ] ) )
    restrict ;
proto proto | all aspath aspath_regexp
    origin any | ( ( [ IGP ] [ EGP ] [ incomplete ] ) )
    [ metric metric ] {
        route_filter [ restrict | ( metric metric ) ] ;
    }
```

When BGP is configured, all routes are assigned an AS path when they are added to the routing table. For all interior routes, this AS path specifies IGP as the origin and no ASEs in the AS path (the current AS is
added when the route is exported). For EGP routes this AS path specifies EGP as the origin and the source AS as the AS path. For BGP routes, the AS path is stored as learned from BGP.

AS path regular expressions are documented in the section on Matching AS paths.

Exporting by Route Tag

```
proto proto | all tag tag restrict ;
proto proto | all tag tag
    [ metric metric ] {
        route_filter [ restrict | ( metric metric ) ] ;
    } ;
```

Both OSPF and RIP version 2 currently support tags; all other protocols always have a tag of zero. The source of exported routes may be selected based on this tag. This is useful when routes are classified by a tag when they are exported into a given routing protocol.

Route Aggregation

Route aggregation is a method of generating a more general route given the presence of a specific route. It is used, for example, at an autonomous system border to generate a route to a network to be advertised via EGP given the presence of one or more subnets of that network learned via RIP. No aggregation is performed unless explicitly requested in an aggregate statement.

Route aggregation is also used by regional and national networks to reduce the amount of routing information passed around. With careful allocation of network addresses to clients, regional networks can just announce one route to regional networks instead of hundreds.

Aggregate routes are not actually used for packet forwarding by the originator of the aggregate route, only by the receiver (if it wishes).

A slight variation of aggregation is the generation of a route based on the existence of certain conditions. This is sometimes known as the route of last resort. This route inherits the next hops and aspath from the contributor specified with the lowest (most favorable) preference. The most common usage for this is to generate a default based on the presence of a route from a peer on a neighboring backbone.

Aggregation and Generation syntax

```
aggregate default
    | ( network [( mask mask ) | ( masklen number ) ] )
    [ preference preference ] [ brief ] {
        proto [ all | direct | static | kernel | aggregate | proto ]
            [ ( as autonomous system ) | ( tag tag )
                | ( aspath aspath_regexp ) ]
        restrict ;
        proto [ all | direct | static | kernel | aggregate | proto ]
            [ ( as autonomous system ) | ( tag tag )
                | ( aspath aspath_regexp ) ]
        [ preference preference ] {
            route_filter [ restrict | ( preference preference ) ] ;
        } ;
    } ;
```

```
generate default
    | ( network [( mask mask ) | ( masklen number ) ] )
    [ preference preference ] {
        [ ( as autonomous system ) | ( tag tag )
            | ( aspath aspath_regexp ) ]
```
Routes that match the route filters are called contributing routes. They are ordered according to the aggregation preference that applies to them. If there are more than one contributing routes with the same aggregating preference, the route’s own preferences are used to order the routes. The preference of the aggregate route will be that of contributing route with the lowest aggregate preference.

**Preference**

Specifies the preference to assign to the resulting aggregate route. The default preference is 130.

**brief**

Used to specify that the AS path should be truncated to the longest common AS path. The default is to build an AS path consisting of SETs and SEQUENCEs of all contributing AS paths.

**proto proto**

In addition to the special protocols listed, the contributing protocol may be chosen from among any of the ones supported (and currently configured into) gated.

**as autonomous_system**

Restrict selection of routes to those learned from the specified autonomous system.

**tag tag**

Restrict selection of routes to those with the specified tag.

**aspath aspath_regexp**

Restrict selection of routes to those that match the specified AS path.

**restrict**

Indicates that these routes are not to be considered as contributors of the specified aggregate. The specified protocol may be any of the protocols supported by the gated daemon. See the section on Route Filters for more detail.

A route may only contribute to an aggregate route that is more general than itself; it must match the aggregate under its mask. Any given route may only contribute to one aggregate route, which will be the most specific configured, but an aggregate route may contribute to a more general aggregate.

**Route Filters**

All the formats allow route filters as shown below. See the section on route filters for a detailed explanation of how they work. When no route filtering is specified (that is, when restrict is specified on the first line of a statement), all routes from the specified source will match that statement. If any filters are specified, only routes that match the specified filters will be considered as contributors. Put differently, if any filters are specified, an all restrict ; is assumed at the end of the list.

**Preference**

Preference is the value the gated daemon uses to order preference of routes from one protocol or peer over another. Preference can be set in the gated.conf configuration file in several different configuration statements.

Preference can be set based on network interface over another, from one protocol over another, or from one remote gateway over another.
Preference may not be used to control the selection of routes within an **IGP**, this is accomplished automatically by the protocol based on metric. Preference may be used to select routes from the same **EGP** learned from different peers or autonomous systems.

Each route has only one preference value associated with it, even though preference can be set at many places in the configuration file. Simply, the last or most specific preference value set for a route is the value used. The preference value is an arbitrarily assigned value used to determine the order of routes to the same destination in a single routing database. The active route is chosen by the lowest preference value.

Some protocols implement a second preference (preference2), sometimes referred to as a tie-breaker.

### Selecting a Route

- The route with the best (numerically smallest) preference is preferred.
- If the two routes have the same preference, the route with the best (numerically smallest) preference2 (also known as a tie-breaker) is preferred.
- A route learned from a **IGP** is preferred to a route learned from an **EGP**. Least preferred is a route learned indirectly by an **IGP** from an **EGP**.
- If AS path information is available it is used to help determine the most preferred route.
  - A route with an AS path is preferred over one without an AS path.
  - If the AS paths and origins are identical, the route with the lower metric is preferred.
  - A route with an AS path origin of **IGP** is preferred over a route with an AS path origin of **EGP**. Least preferred is an AS path with an unknown origin.
- A route with a shorter AS path is preferred.
- If both routes are from the same protocol and AS, the one with the lowest metric is preferred.
- The route with the lowest numeric next-hop address is used.

### Assigning Preferences

A default preference is assigned to each source from which the **gated** daemon receives routes. Preference values range from 0 to 255 with the lowest number indicating the most preferred route.

The following table summarizes the default preference values for routes learned in various ways. The table lists the statements (some of these are clauses within statements) that set preference, and shows the types of routes to which each statement applies. The default preference for each type of route is listed, and the table notes preference precedence between protocols. The narrower the scope of the statement, the higher precedence its preference value is given, but the smaller the set of routes it affects.

<table>
<thead>
<tr>
<th>Preference Of Route</th>
<th>Defined by Statement</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>direct connected networks</td>
<td>interface</td>
<td>0</td>
</tr>
<tr>
<td>OSPF routes</td>
<td>ospf</td>
<td>10</td>
</tr>
<tr>
<td>IS-IS level 1 routes</td>
<td>isis level 1</td>
<td>15</td>
</tr>
<tr>
<td>IS-IS level 2 routes</td>
<td>isis level 2</td>
<td>18</td>
</tr>
<tr>
<td>internally generated default</td>
<td>gendefault</td>
<td>20</td>
</tr>
<tr>
<td>redirects</td>
<td>redirect</td>
<td>30</td>
</tr>
<tr>
<td>routes learned via route socket</td>
<td>kernel</td>
<td>40</td>
</tr>
<tr>
<td>static routes from config</td>
<td>static</td>
<td>60</td>
</tr>
<tr>
<td>ANS SPF (SLSP) routes</td>
<td>slsp</td>
<td>70</td>
</tr>
<tr>
<td>HELLO routes</td>
<td>hello</td>
<td>90</td>
</tr>
<tr>
<td>RIP routes</td>
<td>rip</td>
<td>100</td>
</tr>
<tr>
<td>point-to-point interface</td>
<td>interfaces</td>
<td>110</td>
</tr>
<tr>
<td>routes to interfaces that are down</td>
<td>aggregate/generate</td>
<td>120</td>
</tr>
<tr>
<td>aggregate/generate routes</td>
<td>aggregate/generate</td>
<td>130</td>
</tr>
<tr>
<td>OSPF AS external routes</td>
<td>ospf</td>
<td>150</td>
</tr>
<tr>
<td>BGP routes</td>
<td>bgp</td>
<td>170</td>
</tr>
<tr>
<td>EGP</td>
<td>EGP</td>
<td>200</td>
</tr>
</tbody>
</table>
Sample Preference Specifications

```
interfaces {
  interface 138.66.12.2 preference 10;
}
rip yes {
  preference 90;
}
import proto rip gateway 138.66.12.1 preference 75;
```

In these statements, the preference applicable to routes learned via RIP from gateway 138.66.12.1 is 75. The last preference applicable to routes learned via RIP from gateway 128.66.12.1 is defined in the accept statement. The preference applicable to other RIP routes is found in the rip statement. The preference set on the interface statement applies only to the route to that interface.

The Router Discovery Protocol

The Router Discovery Protocol is an IETF standard protocol used to inform hosts of the existence of routers. It is used in place of, or in addition to statically configured default routes in hosts.

The protocol is split into two portions, the server portion which runs on routers, and the client portion that runs on hosts. The gated daemon treats these much like two separate protocols, only one of which may be enabled at a time.

The Router Discovery Server

The Router Discovery Server runs on routers and announces their existence to hosts. It does this by periodically multicasting or broadcasting a Router Advertisement to each interface on which it is enabled. These Router Advertisements contain a list of all the routers addresses on a given interface and their preference for use as default routers.

Initially, these Router Advertisements occur every few seconds, then fall back to every few minutes. In addition, a host may send a Router Solicitation to which the router will respond with a unicast Router Advertisement (unless a multicast or broadcast advertisement is due momentarily).

Each Router Advertisement contains an Advertisement Lifetime field indicating for how long the advertised addresses are valid. This lifetime is configured such that another Router Advertisement will be sent before the lifetime has expired. A lifetime of zero is used to indicate that one or more addresses are no longer valid.

The Router Advertisements are by default sent to the all-hosts multicast address 224.0.0.1. However, the use of broadcast may be specified. When Router Advertisements are being sent to the all-hosts multicast address, or an interface is configured for the limited-broadcast address 255.255.255.255, all IP addresses configured on the physical interface are included in the Router Advertisement. When the Router Advertisements are being sent to a net or subnet broadcast, only the address associated with that net or subnet is included.

The Router Discovery Server Statement

```
routerdiscovery server yes | no | on | off [ {
  traceoptions trace_options;
  interface interface_list[
    minadvinterval time
    maxadvinterval time
    lifetime time
  ];
  address interface_list
    [advertise] | [ignore]
    [broadcast] | [multicast]
```
traceoptions `trace_options`
Specifies the Router Discovery tracing options. (See Trace Statements and the Router Discovery specific tracing options below.)

`interface `interface_list``
Specifies the parameters that apply to physical interfaces. Note a slight difference in convention from the rest of the gated daemon, `interface` specifies just physical interfaces (such as en0 and tr0), while `address` specifies protocol (in this case IP) addresses.

Interface parameters are:

`maxadvinterval time`
The maximum time allowed between sending broadcast or multicast Router Advertisements from the interface. Must be no less than 4 and no more than 30:00 (30 minutes or 1800 seconds). The default is 10:00 (10 minutes or 600 seconds).

`minadvinterval time`
The minimum time allowed between sending unsolicited broadcast or multicast Router Advertisements from the interface. Must be no less than 3 seconds and no greater than `maxadvinterval`. The default is 0.75 * `maxadvinterval`.

`lifetime time`
The lifetime of addresses in a Router Advertisement. Must be no less than `maxadvinterval` and no greater than 2:30:00 (two hours, thirty minutes or 9000 seconds). The default is 3 * `maxadvinterval`.

`address `interface_list``
Specifies the parameters that apply to the specified set of addresses on this physical interface. Note a slight difference in convention from the rest of `gated.conf`; `interface` specifies just physical interfaces (such as en0 and tr0), while `address` specifies protocol (in this case IP) addresses.

`advertise`
Specifies that the specified address(es) should be included in Router Advertisements. This is the default.

`ignore`
Specifies that the specified address(es) should not be included in Router Advertisements.

`broadcast`
Specifies that the given address(es) should be included in a broadcast Router Advertisement because this system does not support IP multicasting, or some hosts on attached network do not support IP multicasting. It is possible to mix addresses on a physical interface such that some are included in a broadcast Router Advertisement and some are included in a multicast Router Advertisement. This is the default if the router does not support IP multicasting.

`multicast`
Specifies that the given address(es) should only be included in a multicast Router Advertisement. If the system does not support IP multicasting the address(es) will not be included. If the system supports IP multicasting, the default is to include the address(es) in a multicast Router Advertisement if the given interface supports IP multicasting, if not the address(es) will be included in a broadcast Router Advertisement.

`preference preference`
The preferability of the address(es) as a default router address, relative to other router addresses on the same subnet. A 32-bit, signed, twos-complement integer, with higher values meaning more preferable. Note that hex 80000000 may only be specified as `ineligible`. The default is 0.
ineligible
Specifies that the given address(es) will be assigned a preference of (hex 80000000) that means that it is not eligible to be the default route for any hosts.

This is useful when the address(es) should not be used as a default route, but are given as the next hop in an ICMP redirect. This allows the hosts to verify that the given addresses are up and available.

The Router Discovery Client
A host listens for Router Advertisements via the all-hosts multicast address (224.0.0.2), if IP multicasting is available and enabled, or on the interface’s broadcast address. When starting up, or when reconfigured, a host may send a few Router Solicitations to the all-routers multicast address, 224.0.0.2, or the interface’s broadcast address.

When a Router Advertisement with non-zero lifetime is received, the host installs a default route to each of the advertised addresses. If the preference ineligible, or the address is not on an attached interface, the route is marked unusable but retained. If the preference is usable, the metric is set as a function of the preference such that the route with the best preference is used. If more than one address with the same preference is received, the one with the lowest IP address will be used. These default routes are not exportable to other protocols.

When a Router Advertisement with a zero lifetime is received, the host deletes all routes with next-hop addresses learned from that router. In addition, any routers learned from ICMP redirects pointing to these addresses will be deleted. The same will happen when a Router Advertisement is not received to refresh these routes before the lifetime expires.

The Router Discovery Client Statement

```plaintext
routerdiscovery client yes | no | on | off [ { 
  traceoptions trace_options ;
  preference preference ;
  interface [interface_list]
    [ enable ] | [ disable ]
    [ broadcast ] | [ multicast ]
    [ quiet ] | [ solicit ]
  ;
} ] ;
```

traceoptions trace_options
Specifies the tracing options for Router Discovery Client. (See Trace Statements and the Router Discovery Client specific tracing options below.)

preference preference ;
Specifies the preference of all Router Discovery default routes. The default is 55.

interface [interface_list]
Specifies the parameters that apply to physical interfaces. Note a slight difference in convention from the rest of gated, interface specifies just physical interfaces (such as en0 and tr0). The Router Discovery Client has no parameters that apply only to interface addresses.

enable
Specifies that Router Discovery should be performed on the specified interface(s). This is the default.

disable
Specifies that Router Discovery should not be performed on the specified interface(s).
broadcast
Specifies that Router Solicitations should be broadcast on the specified interface(s). This is the default if IP multicast support is not available on this host or interface.

multicast
Specifies that Router Solicitations should be multicast on the specified interface(s). If IP multicast is not available on this host and interface, no solicitation will be performed. The default is to multicast Router Solicitations if the host and interface support it, otherwise Router Solicitations are broadcast.

quiet
Specifies that no Router Solicitations will be sent on this interface, even though Router Discovery will be performed.

solicit
Specifies that initial Router Solicitations will be sent on this interface. This is the default.

Tracing options
The Router Discovery Client and Server support the state trace flag that traces various protocol occurrences.

state  State transitions

The Router Discovery Client and Server do not directly support any packet tracing options, tracing of router discovery packets is enabled via the ICMP Statement.

Route Filtering
Routes are filtered by specifying configuration language that will match a certain set of routes by destination, or by destination and mask. Among other places, route filters are used on martians, import and export statements.

The action taken when no match is found is dependent on the context, for instance import and export route filters assume an all reject; at the end of a list.

A route will match the most specific filter that applies. Specifying more than one filter with the same destination, mask and modifiers will generate an error.

Filtering syntax

network [ exact | refines ]
network mask mask [ exact | refines ]
network masklen number [ exact | refines ]
all
default
host host

These are all the possible formats for a route filter. Not all of these formats are available in all places, for instance the host and default formats are not valid for martians.

In most cases it is possible to specify additional parameters relevant to the context of the filter. For example, on a martian statement it is possible to specify the allow keyword, on an import statement you can specify a preference, and on an export you can specify a metric.

network [ exact | refines ]
network mask mask [ exact | refines ]
network masklen number [ exact | refines ]

Matching usually requires both an address and a mask, although the mask is implied in the shorthand forms listed below. These three forms vary in how the mask is specified. In the first form, the mask is implied to be the natural mask of the network. In the second, the mask is explicitly specified. In the third, the mask is specified by the number of contiguous one bits.
If no additional parameters are specified, any destination that falls in the range given by the network and mask is matched, the mask of the destination is ignored. If a natural network is specified, the network, any subnets, and any hosts will be match. The two optional modifiers cause the mask of the destination to be considered also:

- **exact**  This parameter specifies that the mask of the destination must match the supplied mask exactly. This is used to match a network, but no subnets or hosts of that network.

- **refines**  Specifies that the mask of the destination must be more specified (that is, longer) than the filter mask. This is used to match subnets and/or hosts of a network, but not the network.

**all**  This entry matches anything. It is equivalent to:

```
0.0.0.0 mask 0.0.0.0
```

**default**  Matches the default route. To match, the address must be the default address and the mask must be all zeros. This is equivalent to:

```
0.0.0.0 mask 0.0.0.0 exact
```

- **host**  Matches the specific host. To match, the address must exactly match the specified host and the network mask must be a host mask (that is, all ones). This is equivalent to:

```
host mask 255.255.255.255 exact
```

**Matching AS Paths**

An AS path is a list of autonomous systems that routing information has passed through to get to this router, and an indicator of the origin of the AS path. This information can be used to prefer one path to a destination network over another. The primary method for doing this with gated.conf is to specify a list of patterns to be applied to AS paths when importing and exporting routes.

Each autonomous system that a route passed through prepends its AS number to the beginning of the AS path.

The origin information details the completeness of AS path information. An origin of IGP indicates the route was learned from an interior routing protocol and is most likely complete. An origin of EGP indicates the route was learned from an exterior routing protocol that does not support AS paths (EGP, for example) and the path is most likely not complete. When the path information is definitely not complete, an origin of incomplete is used.

AS path regular expressions are defined in RFC 1164 section 4.2.

**AS Path Matching Syntax**

An AS path is matched using the following syntax:

```
aspath aspath_regexp origin any | ( [IGP] [EGP] [incomplete] )
```

This specifies that an AS matching the `aspath_regexp` with the specified origin is matched.

**AS Path Regular Expressions**

Technically, an AS path regular expression is a regular expression with the alphabet being the set of AS numbers. An AS path regular expression is composed of one or more AS paths expressions. An AS path expressions is composed of AS path terms and AS path operators.

**AS Path Terms**

An AS path term is one of the following three objects:
autonomous_system
.
( aspath_regexp )

autonomous_system
.
( aspath_regexp )

Is any valid autonomous system number, from one through 65534 inclusive.
Matches any autonomous system number.
Contains parentheses group subexpressions—an operator, such as * or ? works on a single element or on a regular expression enclosed in parentheses.

AS Path Operators
An AS path operator is one of the following:

\begin{align*}
&\text{aspath_term} \{m,n\} \\
&\text{aspath_term} \{m\} \\
&\text{aspath_term} \{m,\} \\
&\text{aspath_term} * \\
&\text{aspath_term} + \\
&\text{aspath_term} ? \\
&\text{aspath_term} | \text{aspath_term}
\end{align*}

\text{aspath_term} \{m,n\} \quad \text{a regular expression followed by } \{m,n\} \text{ (where } m \text{ and } n \text{ are both non-negative integers and } m \leq n \text{ means at least } m \text{ and at most } n \text{ repetitions.}
\text{aspath_term} \{m\} \quad \text{a regular expression followed by } \{m\} \text{ (where } m \text{ is a positive integer) means exactly } m \text{ repetitions.}
\text{aspath_term} \{m,\} \quad \text{a regular expression followed by } \{m,\} \text{ (where } m \text{ is a positive integer) means } m \text{ or more repetitions.}
\text{aspath_term} * \quad \text{an AS path term followed by } * \text{ means zero or more repetitions. This is shorthand for } \{0,*\}.
\text{aspath_term} + \quad \text{a regular expression followed by } + \text{ means one or more repetitions.
This is shorthand for } \{1,*\}.
\text{aspath_term} ? \quad \text{a regular expression followed by } ? \text{ means zero or one repetition.
This is shorthand for } \{0,1\}.
\text{aspath_term} | \text{aspath_term} \quad \text{matches the AS term on the left, or the AS term on the right.}

The general format of an entry (contained on a single line) in the \texttt{gateways} file is:

\begin{itemize}
  \item [Destination] A keyword that indicates whether the route is to a network or a specific host. The two possible keywords are \texttt{net} and \texttt{host}.
  \item [Name1] The name associated with \texttt{Destination}. The \texttt{Name1} variable can be either a symbolic name (as used in the \texttt{/etc/hosts} or \texttt{/etc/networks} file) or an Internet address specified in dotted-decimal format.
\end{itemize}
gateway

An indicator that the following string identifies the gateway host.

Name2

The name or address of the gateway host to which messages should be forwarded.

metric

An indicator that the next string represents the hop count to the destination host or network.

Value

The hop count, or number of gateways from the local network to the destination network.

Type

A keyword that indicates whether the gateway should be treated as active, passive, or external. The three possible keywords are:

active

An active gateway is treated like a network interface. That is, the gateway is expected to exchange Routing Information Protocol (RIP) information. As long as the gateway is active, information about it is maintained in the internal routing tables. This information is included with any routing information transmitted through RIP. If the gateway does not respond for a period of time, the associated route is deleted from the internal routing tables.

passive

A passive gateway is not expected to exchange RIP information. Information about the gateway is maintained in the routing tables indefinitely and is included with any routing information transmitted through RIP.

external

An external gateway is identified to inform the routed daemon that another routing process will install such a route and that alternative routes to that destination should not be installed. Information about external gateways is not maintained in the internal routing tables and is not transmitted through RIP.

Note: These routes must be to networks.

Examples

1. To specify a route to a network through a gateway host with an entry in the gateways file, enter a line in the following format:

   net net2 gateway host4 metric 4 passive

   This example specifies a route to a network, net2, through the gateway host4. The hop count metric to net2 is 4 and the gateway is treated as passive.

2. To specify a route to a host through a gateway host with an entry in the gateways file, enter a line in the following format:

   host host2 gateway host4 metric 4 passive

   This example specifies a route to a host, host2, through the gateway host4. The hop count metric to host2 is 4 and the gateway is treated as passive.

3. To specify a route to a host through an active Internet gateway with an entry in the gateways file, enter a line in the following format:

   host host10 gateway 192.100.11.5 metric 9 active

   This example specifies a route to a specific host, host10, through the gateway 192.100.11.5. The hop count metric to host10 is 9 and the gateway is treated as active.

4. To specify a route to a host through a passive Internet gateway with an entry in the gateways file, enter a line in the following format:

   host host10 gateway 192.100.11.5 metric 9 passive

5. To specify a route to a network through an external gateway with an entry in the gateways file, enter a line in the following format:

   net net5 gateway host7 metric 11 external

   This example specifies a route to a network, net5, through the gateway host7. The hop count metric to net5 is 11 and the gateway is treated as external (that is, it is not advertised through RIP but instead through an unspecified routing protocol).
Files

/usr/lpp/tcpip/samples/gateways Contains the sample gateways file, which also contains directions for its use.

Related Information
The routed daemon.

TCP/IP routing gateways TCP/IP protocols TCP/IP routing in Networks and communication management.

hosts File Format for TCP/IP

Purpose
Defines the Internet Protocol (IP) name and address of the local host and specifies the names and addresses of remote hosts.

Description
The /etc/hosts file contains the Internet Protocol (IP) host names and addresses for the local host and other hosts in the Internet network. This file is used to resolve a name into an address (that is, to translate a host name into its Internet address). When your system is using a name server, the file is accessed only if the name server cannot resolve the host name.

When the local host is using the DOMAIN protocol, the resolver routines query a remote DOMAIN name server before searching this file. In a flat network with no name server, the resolver routines search this file for host name and address data.

Entries in the hosts file have the following format:

Address HostName

In this entry, Address is an IP address specified in either dotted decimal or octal format, and HostName is the name of a host specified in either relative or absolute domain name format. If you specify the absolute domain name, the portion of the name preceding the first . (period) has a maximum length of 63 characters and cannot contain blanks. For both formats of the name, the total number of characters cannot exceed 255 characters, and each entry must be contained on one line. Multiple HostNames (or aliases) can be specified.

Note: Valid host names or alias host names must contain at least one alphabetic character. If you choose to specify a host name or alias that begins with an x followed by any hexadecimal digit (0-f), the host name or alias must also contain at least one additional letter that cannot be expressed as a hexadecimal digit. The system interprets a leading x followed by a hexadecimal digit as the base 16 representation of an address, unless there is at least one character in the host name or alias that is not a hexadecimal digit. Thus, xdeer would be a valid host name, whereas xdee would not.

This file can contain two special case entries that define reserved (or well-known) host names. These host names are:

timeserver Identifies a remote time server host. This host name is used by the setclock command.
printserver Identifies the default host for receiving print requests.
In this **hosts** file entry, the *Address* parameter is an IP address specified in either dotted decimal or octal format, and each *HostName* parameter is a host name specified in either relative or absolute domain name format. These never have the full domain name listed; they are always listed as either printserver or timeserver.

**Note:** The local `/etc/resolv.conf` file defines where DOMAIN name servers are, and the name server file defines where Internet services are available. Although it is not necessary to define well-known hosts in the **hosts** file when using the DOMAIN protocol, it may be useful if they are not defined by your name server.

Entries in this file can be made by using the System Management Interface Tool (SMIT) or the **hostent** command, or by creating and editing the file with an editor.

### Examples

In these examples, the name of the local host is the first line in each **hosts** file. This is to help you identify the host whose file is being displayed. Your host does not have to be defined on the first line of your **hosts** file.

1. The following sample entries might be contained in the **hosts** files for two different hosts on a network that is not running a DOMAIN name server:

   **Host1**
   
   185.300.10.1 host1
   185.300.10.2 host2
   185.300.10.3 host3
   185.300.10.4 host4 merlin
   185.300.10.5 host5 arthur king
   185.300.10.5 timeserver

   **Host 2**
   
   185.300.10.2 host2
   185.300.10.1 host1
   185.300.10.3 host3
   185.300.10.4 host4 merlin
   185.300.10.5 host5 arthur king

   In this sample network with no name server, the **hosts** file for each host must contain the Internet address and host name for each host on the network. Any host that is not listed cannot be accessed. The host at Internet address 185.300.10.4 in this example can be accessed by either name: host4 or merlin. The host at Internet address 185.300.10.5 can be accessed by any of the names host5, arthur, or king.

2. Following is a sample entry in the **hosts** files for a different host on a DOMAIN network, but the host is not the name server, and the host is keeping some additional host names for a smaller network:

   **Host 5**
   
   128.114.1.15 name1.xyz.aus.century.com name1
   128.114.1.14 name2.xyz.aus.century.com name2
   128.114.1.16 name3.xyz.aus.century.com name3

   In this sample, host5 is not a name server, but is attached to a DOMAIN network. The hosts file for host5 contains address entries for all hosts in the smaller network, and the DOMAIN data files contain the DOMAIN database. The entries in the **hosts** file that begin with 128.114 indicate that host5 resolves names for hosts on the smaller network.

### Related Information

The **hostent** command, **setclock** command.

The **getlinebyaddr** routine.
hosts.equiv File Format for TCP/IP

Purpose
Specifies remote systems that can execute commands on the local system.

Description
The `/etc/hosts.equiv` file, along with any local `$HOME/.rhosts` files, defines the hosts (computers on a network) and user accounts that can invoke remote commands on a local host without supplying a password. A user or host that is not required to supply a password is considered trusted.

When a local host receives a remote command request, the appropriate local daemon first checks the `/etc/hosts.equiv` file to determine if the request originates with a trusted user or host. For example, if the local host receives a remote login request, the `rlogin` daemon checks for the existence of a `hosts.equiv` file on the local host. If the file exists but does not define the host or user, the system checks the appropriate `$HOME/.rhosts` file. This file is similar to the `/etc/hosts.equiv` file, except that it is maintained for individual users.

Both files, `hosts.equiv` and `.rhosts` must have permissions denying write access to group and other. If either group or other have write access to a file, that file will be ignored.

Do not give write permission to the `/etc/hosts.equiv` file to group and others. Permissions of the `/etc/hosts.equiv` file should be set to 600 (read and write by owner only).

If a remote command request is made by the root user, the `/etc/hosts.equiv` file is ignored and only the `/rhosts` file is read.

Note: Be careful when establishing trusted relationships. Networks that use trusted facilities can be less secure than those that do not.

Granting and Denying Trust
You grant trust from a local host to a remote host or remote user. The local machine's `/etc/hosts.equiv` file contains entries for each trusted host or user. The format of an entry is:

```
HostName [UserName]
```

The `HostName` field specifies the name of the host to trust. The `UserName` field specifies the name of the user on that remote host to trust. The `UserName` field is optional.

You can use the + (plus sign) as a wildcard in either the `HostName` or `UserName` field to grant trust to all users from a particular host or from all hosts that a specific user has an account on. To grant trust to every user on every machine on the network, place a plus sign (+) at the beginning of the file.

Note: When granting access through the `/etc/hosts.equiv` file, extreme caution must be used. Lines that include a `UserName`, either as an individual user, a netgroup, or the + (plus sign used as a wildcard character), permit the qualifying users to access the system as any non-root local user.

You deny a host or user trust by omitting them from the `/etc/hosts.equiv` file altogether. By omitting the host or user, you imply they are not trusted. This is the most secure way to deny trust. Otherwise, you can explicitly deny trust to a specific host or user by using the - (minus sign). The format to explicitly deny a host is:

```
-HostName
```
The format to explicitly deny a specific user from a host is:

\[
\text{HostName~[\neg UserName]}\]

**Using NIS with the /etc/hosts.equiv file**

If your network uses the Network Information Services (NIS), you can use netgroups in place of either the *HostName* or *UserName* field. The system resolves the netgroup depending on which field the netgroup replaces. For example, if you place a netgroup in the *HostName* field, the system resolves the hosts component of the netgroup. If the netgroup appears in the *UserName* field, the user component is resolved. Use the following format to grant trust to a netgroup:

\[
+@\text{NetGroup}\]

To deny trust, use the following:

\[
-@\text{NetGroup}\]

Refer to the NIS [netgroup] file for more information on netgroups.

**Ordering Entries in the /etc/hosts.equiv File**

The order of entries in the /etc/hosts.equiv file is important. When verifying trust, the system parses the /etc/hosts.equiv file from top to bottom. When it encounters an entry that matches the host or user attempting a remote command, the system stops parsing the file and grants or denies trust based on the entry. Any additional entries that appear later in the file are ignored.

**Examples**

1. To allow all the users on remote hosts emerald and amethyst to log in to host diamond, enter:

   emerald
   amethyst

   These entries in diamond’s /etc/hosts.equiv file allow all the users on emerald and amethyst with local accounts on diamond to remotely log in without supplying a password.

2. To allow only the user gregory to remotely login to diamond from host amethyst, enter:

   emerald
   amethyst gregory

   This entry in diamond’s /etc/hosts.equiv file forces all the users on amethyst, except for gregory, to supply a password when remotely logging in to diamond.

3. To grant trust to peter regardless of the host he attempts to execute remote commands from, enter:

   emerald
   amethyst gregory
   + peter

   This entry in diamond’s /etc/hosts.equiv file allows peter to execute remote commands on diamond from any host that he has an account on.

4. To allow all hosts in the century netgroup to execute remote commands on host diamond, enter:

   emerald
   amethyst gregory
   + peter
   +@century

   This entry in diamond’s /etc/hosts.equiv file grants trust to all hosts in the century netgroup. This means that any user with an account on a century host and an account on diamond can execute remote commands on diamond without supplying a password.

5. To allow all the users in the engineers netgroup with accounts on citrine to execute remote commands on host diamond, enter:
6. To grant trust to all users with accounts on hosts in the servers netgroup that are users in the sysadmins netgroup, enter:

   emerald
   amethyst gregory
   + peter
   +@century
citrine +@engineers
   +@servers +@sysadmins

   This entry in diamond's /etc/hosts.equiv file grants trust to any user in the sysadmins netgroup who is remotely executing commands from hosts that are in the servers netgroup.

7. To force an engineers netgroup user lydia who has an account on citrine to use a password while allowing all other engineers users not to, enter:

   emerald
   amethyst gregory
   + peter
   +@century
citrine -lydia
   citrine +@engineers
   +@servers +@sysadmins

   This entry in diamond's /etc/hosts.equiv file grants trust to all of netgroup engineers users, except for lydia, who must supply a password. The order of entries is very important. Recall that the system grants trust based on the first entry it encounters. If the order of the entries appeared as follows:

   emerald
   amethyst gregory
   + peter
   +@century
citrine +@engineers
   +@servers +@sysadmins
   citrine -lydia
   +@servers +@sysadmins

   User lydia, as a member of engineers, would be allowed to execute remote commands on diamond even though a later entry explicitly denies her trust.

Files

$HOME/.rhosts Specifies remote users who can use a local-user account.

Related Information

The NIS netgroup file.

The TCP/IP .rhosts file format.

The lpd command, rcp command, rdist command, rdump command, rlogin command, rsh command, ruser command.

The rlogind daemon, rshd daemon.
hosts.lpd File Format for TCP/IP

Purpose
Specifies remote hosts that can print on the local host.

Description
The /etc/hosts.lpd file defines which remote systems are permitted to print on the local system. The remote systems listed in this file do not have the full privileges given to files listed in the /etc/hosts.equiv file.

Host-Name Field
The hosts.lpd file supports the following host-name entries:

+ HostName
-HostName
+@NetGroup
-@NetGroup

A + (plus sign) signifies that any host on the network can print using the local host. The HostName entry is the name of a remote host and signifies that HostName can print, using the local host. A -HostName entry signifies the host is not allowed to print using the local host. A +@NetGroup or -@NetGroup entry signifies all hosts in the netgroup or no hosts in the netgroup, respectively, are allowed to print using the local host.

The @NetGroup parameter is used by Network Information Service (NIS) for grouping. Refer to the NIS netgroup file for more information on netgroups.

Entries in this file can be made using the System Management Interface Tool (SMIT) or the ruser command.

Note: Comments must be entered on separate lines in the hosts.lpd file. Comments should not be entered on lines containing host names.

To implement hosts.lpd file changes without restarting the system, use the System Resource Controller (SRC) refresh command.

Examples
1. To allow remote specified hosts to print using a local host, enter:
   hamlet
   lear
   prospero
   setebos

   These entries in the local host's /etc/hosts.lpd file allow hosts hamlet, lear, prospero, and setebos to print files, using the local host.

2. To prevent a remote host from printing using a local host, enter:
   -hamlet

   This entry in the local host's /etc/hosts.lpd file prevents host hamlet from printing files, using the local host.

3. To allow all hosts in an NIS netgroup to print using the local host, enter:
   +@century
This entry in the local host's `/etc/hosts.lpd` file allows all hosts in the `century` netgroup to print files, using the local host. The `@` (at sign) signifies the network is using NIS grouping.

### Files

`/etc/hosts.equiv` Specifies remote systems that can execute commands on the local system.

### Related Information

- The `netgroup` file for NIS.
- The `hosts.equiv` file format for TCP/IP.
- The `lpd` command, `ruser` command.

[Managing and Using Remote Printers and Queues](#) and [Remote Printing Overview](#) in *Printers and printing*.

### hty_config File Format

#### Purpose

Specifies the number of hty's to configure on a Network Terminal Accelerator adapter.

#### Description

The `/etc/hty_config` file supplies the `hty_load` command with information to define ports for a specified device. The System Management Interface Tool (SMIT) writes to this file when hty devices are configured, specifying the device by supplying the adapter minor number for the device. Both the number of ports and the device are specified in a three-column table that can have multiple lines.

The Cluster Address column defines the cluster controller's network address. For the boards, the cluster address should be set to 1. Any other value may cause unpredictable results.

After you have configured the Network Terminal Accelerator adapter with SMIT, the `hty_config` file appears similar to the following:

<table>
<thead>
<tr>
<th>Adapter minor #</th>
<th>Cluster address</th>
<th>Number of ports</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>256</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>700</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>85</td>
</tr>
</tbody>
</table>

In this example, the host has three adapters, the first of which is configured for 256 hty devices, the second for 700, and the third for 85.

### Related Information

- The `hty_load` command

### inetd.conf File Format for TCP/IP

#### Purpose

Defines how the `inetd` daemon handles Internet service requests.
Description

The `/etc/inetd.conf` file is the default configuration file for the ` inetd` daemon. This file enables you to specify the daemons to start by default and supply the arguments that correspond to the desired style of functioning for each daemon. This file is part of TCP/IP in Network Support Facilities.

If you change the `/etc/inetd.conf` file, run the `refresh -s inetd` or `kill -1 InetdPID` command to inform the `inetd` daemon of the changes to its configuration file. The `inetd.conf` file specifies which daemons start by default and supplies arguments determining the style of functioning for each daemon.

The following daemons are controlled by the `inetd` daemon:

- `comsat`
- `ftpd`
- `telnetd`
- `rshd`
- `rlogind`
- `rexecd`
- `fingerd`
- `tftpd`
- `talkd`
- `uucpd`

The `ftpd`, `rlogind`, `rexecd`, `rshd`, `talkd`, `telnetd`, and `uucpd` daemons are started by default. The `tftpd`, `fingerd`, and `comsat` daemons are not started by default unless they are uncommented in the `/etc/inetd.conf` file.

Service Requests

The following Internet service requests are supported internally by the `inetd` daemon and are generally used for debugging:

- **ECHO** Returns data packets to a client host.
- **DISCARD** Discards received data packets.
- **CHARGEN** Discards received data packets and sends predefined or random data.
- **DAYTIME** Sends the current date and time in user-readable form.
- **TIME** Sends the current date and time in machine-readable form.

The `inetd` daemon reads its configuration file only when the `inetd` daemon starts, when the `inetd` daemon receives a `SIGHUP` signal, or when the SRC `refresh -s inetd` command is entered. Each line in the `inetd` configuration file defines how to handle one Internet service request only.

Each line is of the form:

```
ServiceName  SocketType  ProtocolName Wait/NoWait  UserName ServerPath ServerArgs
```

These fields must be separated by spaces or tabs and have the following meanings:

- **ServiceName** Contains the name of an Internet service defined in the `/etc/services` file. For services provided internally by the `inetd` daemon, this name must be the official name of the service. That is, the name must be identical to the first entry on the line that describes the service in the `/etc/services` file.
SocketType

Contains the name for the type of socket used for the service. Possible values for the SocketType parameter are:

- **stream**: Specifies that a stream socket is used for the service.
- **dgram**: Specifies that a datagram socket is used for the service.
- **sunrpc_tcp**: Specifies that a Sun remote procedure call (RPC) socket is used for the service, over a stream connection.
- **sunrpc_udp**: Specifies that a Sun RPC socket is used for the service, over a datagram connection.

ProtocolName

Contains the name of an Internet protocol defined in the /etc/protocols file. For example, use the tcp value for a service that uses TCP/IP and the udp value for a service that uses the User Datagram Protocol (UDP).

Wait/NoWait

Contains either the wait or the nowait instruction for datagram sockets and the nowait instruction for stream sockets. The Wait/NoWait field determines whether the inetd daemon waits for a datagram server to release the socket before continuing to listen at the socket.

Wait/NoWait/SRC

Contains either the wait, the nowait, or the SRC instruction for datagram sockets and the nowait instruction for stream sockets. The Wait/NoWait/SRC field determines whether the inetd daemon waits for a datagram server to release the socket before continuing to listen at the socket. The SRC instruction works like wait, but instead of forking and waiting for the child to die, it does a startsrc on the subsystem and stores information about the starting of the service. When the service is removed from the inetd.conf file and inetd is restarted, the service then has a stopsrc issued to the service to stop it. If DEBUG[level] is specified after wait or nowait, the SO_DEBUG socket debugging flag will be turned on for this service. If level is specified, the trace level is set to the specified level; otherwise the level is set to normal. Valid values for level are min, normal, or detail.

UserName

Specifies the user name that the inetd daemon should use to start the server. This variable allows a server to be given less permission than the root user.

ServerPath

Specifies the full path name of the server that the inetd daemon should execute to provide the service. For services that the inetd daemon provides internally, this field should be internal.

ServerArgs

Specifies the command line arguments that the inetd daemon should use to execute the server. The maximum number of arguments is five. The first argument specifies the name of the server used. If the SocketType parameter is sunrpc_tcp or sunrpc_udp, the second argument specifies the program name and the third argument specifies the version of the program. For services that the inetd daemon provides internally, this field should be empty.

Examples

The following are example entries in the /etc/inetd.conf file for an inetd daemon that:

- Uses the ftpd daemon for servicing ftp requests
- Uses the talkd daemon for talk requests
- Uses the telnetd daemon for telnet requests, sets the SO_DEBUG flag for sockets used for this service, and sets the trace level for these sockets to normal.
- Provides time requests internally.

```plaintext
ftp&rbl;stream tcp nowait root /usr/sbin/ftpd ftpd
ntalk dgram udp wait root /usr/sbin/talkd talkd
telnet stream tcp6 nowait|DEBUG root /usr/sbin/telnetd telnetd -a
time stream tcp nowait root internal
time dgram udp wait root internal
```

434 AIX 5L Version 5.3 Files Reference
Related Information

The `kill` command, `refresh` command.

The `inetd` daemon.

The `protocols` file format, `services` file format.

`Lastlog File Format`

Purpose

Defines the last login attributes for users.

Description

The `/etc/security/lastlog` file is an ASCII file that contains stanzas with the last login attributes for users. Each stanza is identified by a user name and contains attributes in the `Attribute=Value` form. Each attribute is ended by a new-line character, and each stanza is ended by an additional new-line character.

Each stanza can have the following attributes:

- `time_last_login` Specifies the number of seconds since the epoch (00:00:00 GMT, January 1, 1970) since the last successful login. The value is a decimal integer.
- `tty_last_login` Specifies the terminal on which the user last logged in. The value is a character string.
- `host_last_login` Specifies the host from which the user last logged in. The value is a character string.
- `unsuccessful_login_count` Specifies the number of unsuccessful login attempts since the last successful login. The value is a decimal integer. This attribute works in conjunction with the user's `loginretries` attribute, specified in the `/etc/security/user` file, to lock the user's account after a specified number of consecutive unsuccessful login attempts. Once the user's account is locked, the user will not be able to log in until the system administrator resets the user's `unsuccessful_login_count` attribute to be less than the value of `loginretries`. To do this, enter the following:

```
chsec -f /etc/security/lastlog -s username -a \nunsuccessful_login_count=0
```

- `time_last_unsuccessful_login` Specifies the number of seconds since the epoch (00:00:00 GMT, January 1, 1970) since the last unsuccessful login. The value is a decimal integer.
- `tty_last_unsuccessful_login` Specifies the terminal on which the last unsuccessful login attempt occurred. The value is a character string.
- `host_last_unsuccessful_login` Specifies the host from which the last unsuccessful login attempt occurred. The value is a character string.
All user database files should be accessed through the system commands and subroutines defined for this purpose. Access through other commands or subroutines may not be supported in future releases.

The `mkuser` command creates a user stanza in the `lastlog` file. The attributes of this user stanza are initially empty. The field values are set by the `login` command as a result of logging in to the system. The `lsuser` command displays the values of these attributes; the `rmuser` command removes the user stanza from this file, along with the user account.

**Security**

Access Control: This command should grant read (r) access to the root user, members of the security group, and others consistent with the security policy for the system. Only the root user should have write (w) access.

**Examples**

A typical stanza is similar to the following example for user bck:

```plaintext
bck:
  time_last_unsuccessful_login = 732475345
  tty_last_unsuccessful_login = tty0
  host_last_unsuccessful_login = waterski
  unsuccessful_login_count = 0
  time_last_login = 734718467
  tty_last_login = lft/0
  host_last_login = waterski
```

**Files**

```
/etc/security/lastlog
/etc/group
/etc/security/group
/etc/passwd
/etc/security/passwd
/etc/security/environ
/etc/security/user
/etc/security/limits
```

Specifies the path to the `lastlog` file.
Contains the basic attributes of groups.
Contains the extended attributes of groups.
Contains the basic attributes of users.
Contains password information.
Contains the environment attributes of users.
Contains the extended attributes of users.
Contains the process resource limits of users.

**Related Information**

The `login` command, `lsuser` command, `mkuser` command, `rmuser` command, `su` command.

The `getuserattr` subroutine, `putuserattr` subroutine.

**Idap.cfg File Format**

**Purpose**

The `secldapclntd` LDAP client side daemon configuration file.

**Description**

The `/etc/security/ldap/idap.cfg` file contains information for the `secldapclntd` daemon to start and function properly as well as information for fine tuning the daemon’s performance. The `/etc/security/ldap/idap.cfg` file is updated by the `mksecldap` command at client setup.
The `/etc/security/ldap/ldap.cfg` file may contain the following fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ldapservers</code></td>
<td>Specifies a comma separated list of LDAP Security Information Servers. These servers can either be the primary server and/or replica of the primary server. The first server in the list has the highest priority.</td>
</tr>
<tr>
<td><code>binddn</code></td>
<td>Specifies the DN LDAP used to bind to the LDAP Security Information Server(s).</td>
</tr>
<tr>
<td><code>bindpwd</code></td>
<td>Specifies the password for the <code>binddn</code>.</td>
</tr>
<tr>
<td><code>authtype</code></td>
<td>Specifies the authentication mechanism to use. Valid values are <code>unix_auth</code> and <code>ldap_auth</code>. The default is <code>unix_auth</code>.</td>
</tr>
<tr>
<td></td>
<td><code>unix_auth</code> - Retrieves the user password from LDAP and authenticate the user locally.</td>
</tr>
<tr>
<td></td>
<td><code>ldap_auth</code> - Binds to the LDAP server as the authenticating user in order to authenticate.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Password will be sent in clear text to the LDAP server for <code>ldap_auth</code> authentication mechanism. Use of SSL is strongly encouraged.</td>
</tr>
<tr>
<td><code>useSSL</code></td>
<td>Specifies whether to use SSL communication. Valid values are <code>yes</code> and <code>no</code>. The default is <code>no</code>.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> You will need the SSL key and the password to the key to enable this feature.</td>
</tr>
<tr>
<td><code>ldapsskeyf</code></td>
<td>Specifies the full path to the SSL key.</td>
</tr>
<tr>
<td><code>ldapsskeypwd</code></td>
<td>Specifies the password to the SSL key.</td>
</tr>
<tr>
<td><code>useKRB5</code></td>
<td>Specifies whether to use Kerberos for the initial bind to the server. Valid values are <code>yes</code> or <code>no</code>. The default is <code>no</code>.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> The Kerberos principal, key path and kinit command directory are required to enable this feature. If Kerberos bind is enabled then the <code>binddn</code> and <code>bindpwd</code> are not required.</td>
</tr>
<tr>
<td><code>krbprincipal</code></td>
<td>Specifies the Kerberos principal used to bind to the server.</td>
</tr>
<tr>
<td><code>krbkeypath</code></td>
<td>Specifies the path to the kerberos keytab. The default is <code>/etc/security/ldap/krb5.keytab</code>.</td>
</tr>
<tr>
<td><code>krbcmdmdir</code></td>
<td>Specifies the directory that contains the Kerberos <code>kinit</code> command. The default is <code>/usr/krb5/bin/</code>.</td>
</tr>
<tr>
<td><code>userattrmappath</code></td>
<td>Specifies the full path to the AIX-LDAP attribute map for users.</td>
</tr>
<tr>
<td><code>groupattrmappath</code></td>
<td>Specifies the full path to the AIX-LDAP attribute map for groups.</td>
</tr>
<tr>
<td><code>idattrmappath</code></td>
<td>Specifies the full path to the AIX-LDAP attribute map for IDs. These IDs are used by the <code>mkuser</code> command when creating LDAP users.</td>
</tr>
<tr>
<td><code>userbasedn</code></td>
<td>Specifies the user base DN. See <a href="#">Detailed information</a> for more information.</td>
</tr>
<tr>
<td><code>groupbasedn</code></td>
<td>Specifies the group base DN. See <a href="#">Detailed information</a> for more information.</td>
</tr>
<tr>
<td><code>idbasedn</code></td>
<td>Specifies the ID base DN. See <a href="#">Detailed information</a> for more information.</td>
</tr>
<tr>
<td><code>hostbasedn</code></td>
<td>Specifies the host base DN. See <a href="#">Detailed information</a> for more information.</td>
</tr>
<tr>
<td><code>servicebasedn</code></td>
<td>Specifies the service base DN. See <a href="#">Detailed information</a> for more information.</td>
</tr>
<tr>
<td><code>protocolbasedn</code></td>
<td>Specifies the protocol base DN. See <a href="#">Detailed information</a> for more information.</td>
</tr>
<tr>
<td><code>networkbasedn</code></td>
<td>Specifies the network base DN. See <a href="#">Detailed information</a> for more information.</td>
</tr>
<tr>
<td><code>netgroupbasedn</code></td>
<td>Specifies the netgroup base DN. See <a href="#">Detailed information</a> for more information.</td>
</tr>
<tr>
<td><code>rpcbasedn</code></td>
<td>Specifies the RPC base DN. See <a href="#">Detailed information</a> for more information.</td>
</tr>
<tr>
<td><code>aliasbasedn</code></td>
<td>Specifies the alias base DN. See <a href="#">Detailed information</a> for more information.</td>
</tr>
<tr>
<td><code>automountbasedn</code></td>
<td>Specifies the automount base DN. See <a href="#">Detailed information</a> for more information.</td>
</tr>
<tr>
<td><code>bootparambasedn</code></td>
<td>Specifies the bootparams base DN. See <a href="#">Detailed information</a> for more information.</td>
</tr>
<tr>
<td><code>etherbasedn</code></td>
<td>Specifies the ether base DN. See <a href="#">Detailed information</a> for more information.</td>
</tr>
<tr>
<td><strong>userclasses</strong></td>
<td>Specifies a comma-separated list of object classes that are used for the user entry. See <a href="#">Objectclasses in Detailed information</a> for more information.</td>
</tr>
<tr>
<td><strong>groupclasses</strong></td>
<td>Specifies a comma-separated list of object classes that are used for the group entry. See <a href="#">Objectclasses in Detailed information</a> for more information.</td>
</tr>
<tr>
<td><strong>ldapversion</strong></td>
<td>Specifies the LDAP server protocol version. Default is 3.</td>
</tr>
<tr>
<td><strong>ldapport</strong></td>
<td>Specifies the port that the LDAP server listens to. Default is 389.</td>
</tr>
<tr>
<td><strong>ldapsslport</strong></td>
<td>Specifies the SSL port that the LDAP server listens to. Default is 636.</td>
</tr>
<tr>
<td><strong>followaliases</strong></td>
<td>Specifies whether to follow aliases. Valid values are NEVER, SEARCHING, FINDING, and ALWAYS. Default is NEVER.</td>
</tr>
<tr>
<td><strong>usercachesize</strong></td>
<td>Specifies the user cache size. Valid values are 100 - 10,000 entries. Default is 1,000.</td>
</tr>
<tr>
<td><strong>groupcachesize</strong></td>
<td>Specifies the group cache size. Valid values are 10 - 1,000 entries. Default is 100.</td>
</tr>
<tr>
<td><strong>cachetimeout</strong></td>
<td>Specifies the cache TTL (time to live). Valid values are 60 - 3,600 seconds. Default is 300. Set to 0 to disable caching.</td>
</tr>
<tr>
<td><strong>heartbeatinterval</strong></td>
<td>Specifies the interval in seconds that the client contacts the server for server status. Valid values are 60 - 3,600 seconds. Default is 300.</td>
</tr>
<tr>
<td><strong>numberofthread</strong></td>
<td>Specifies the number of threads for the secldapclntd daemon. Valid values are 1 - 1,000. Default is 10.</td>
</tr>
<tr>
<td><strong>nsorder</strong></td>
<td>Specifies the order of host name resolution by the secldapclntd daemon. The default order is dns, nis, local. For more information about valid resolvers, see <a href="#">TCP/IP Name Resolution</a>. <strong>Note:</strong> Do not use nis_ldap, because it could result in the secldapclntd daemon hang.</td>
</tr>
</tbody>
</table>
| **searchmode** | Specifies the set of user and group attributes to be retrieved. This attribute is intended for use for performance reasons. The AIX commands may not be enabled to support all non-OS attributes. Valid values are ALL and OS. The default is ALL.  
**ALL** - Retrieve all attributes of an entry.  
**OS** - Retrieve only the operating system required attributes of an entry. Non-OS attributes like telephone number, binary images etc. will not be returned.  
**Note:** Only use OS when entries have many non-OS required attributes or attributes with large value, e.g. binary data, to reduce sorting effort by the LDAP server. |
| **defaultentrylocation** | Specifies the location of the default entry. Valid values are ldap and local. The default is ldap.  
**ldap** - Use the default entry in LDAP for all attribute default values.  
**local** - Use the default stanza from local /etc/security/user file for all attribute default values. |
| **ldaptimeout** | Specifies the timeout period in seconds for LDAP client requests to the server. This value determines how long the client will wait for a response from the LDAP server. Valid range is 0 - 3600 (1 hour). Default is 60 seconds. Set this value to 0 to disable the timeout. |
| **connectionsperserver** | Specifies the number of LDAP server connections to maintain for each server in the ldapservers list. If the specified value is greater than numberofthread, then only numberofthread connections are made. Valid values are 1 to 100. The default is 10. |
| **serverschematype** | Specifies the schema type of the LDAP server. It is set by the mksecldap command at LDAP client configuration time. Do not modify this attribute. Valid values are: rfc2307aix, rfc2307, aix, sfu30, and sfur2. |
| **enableutf8_xlation** | Enables the saving of data to the LDAP server in UTF-8 format. Valid values are yes and no. The default value is no. |

**Detailed information**
- Multiple base DNs
All of the base DN attributes accept multiple values, with each `<basedn>: <value>` pair on a separate line. For example, to allow users in the `ou=dept1users,cn=aixdata` base DNs and the `ou=dept2users,cn=aixdata` base DNs to log in to the system, you can specify the `userbasedn` attribute as follows:

```
userbasedn: ou=dept1users,cn=aixdata
userbasedn: ou=dept2users,cn=aixdata
```

You can specify up to 10 base DNs for each entity in the `/etc/security/ldap/ldap.cfg` file. The base DNs are prioritized in the order they appear in the `/etc/security/ldap/ldap.cfg` file. The following list describes the system behaviors in regards to multiple base DNs:

- Query operations, such as the `lsuser` command, are done according to the base DN order that is specified until a matching account is found. A failure is returned only if all of the base DNs are searched without finding a match.
- Modification operations, such as the `chuser` command, are done to the first matching account.
- Deletion operations, such as the `rmuser` command, are done to the first matching account.
- Creation operations, such as the `mkuser` command, are done only to the first base DN.

**Extended base DN format**

You can specify optional parameters of search scope and search filter for base DN attributes. You can append the parameters to the base DN with fields separated by question mark (?) characters. The following list shows the valid base DN formats:

- This format represents the default format that the `secldapclntd` daemon uses:
  ```
  userbasedn: ou=people, cn=aixdata
  ```
- This format limits the search by a scope attribute:
  ```
  userbasedn: ou=people, cn=aixdata?scope
  ```
  The scope attribute accepts the following values:
  - `sub`
  - `one`
  - `base`
  If you do not specify the scope attribute, the default value is `sub`.
- This format limits the search by a filter attribute.
  ```
  userbasedn: ou=people, cn=aixdata??filter
  ```
  The filter attribute limits the entries that are defined in the LDAP server. You can use this filter to make only users with certain properties visible to the system. The following list shows some valid filter formats, where `attribute` is the name of an LDAP attribute, and `value` specifies the search criteria, which can be a wild card (`*`).
  - `(attribute=value)`
  - `(&(attribute=value)(attribute=value))`
  - `((attribute=value)(attribute=value))`
- This format uses both a scope attribute and a filter attribute.
  ```
  userbasedn: ou=people, cn=aixdata?scope?filter
  ```

**Object classes**

The first object class in the list is the key object class, which can be used for search operations. By default, the `keyobjectclass` attribute in the attribute mapping file is used for this purpose. But if the mapping file does not exist, or the `keyobjectclass` attribute is not present in the mapping file, the first object class in this list is used.

**Related Information**

The `mksecldap` command and `secldapclntd` daemon.
The `start-secldapclntd`, `stop-secldapclntd`, `restart-secldapclntd`, `ls-secldapclntd` and `flush-secldapclntd` commands.

The **AIX-LDAP Attribute Mapping File Format** file.

**Lightweight Directory Access Protocol** in *Operating system and device management.*

## LDAP Attribute Mapping File Format

### Purpose

Defines AIX to LDAP attribute name mapping to support configurable LDAP server schema.

### Description

These map files are used by the `/usr/lib/security/LDAP` module and the `secldapclntd` daemon for translation between AIX attribute names to LDAP attribute names. Each entry in a mapping file represents a translation for an attribute. A entry has four space separated fields:

<table>
<thead>
<tr>
<th>AIX_Attribute_Name</th>
<th>Specifies the AIX attribute name.</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIX_Attribute_Type</td>
<td>Specifies the AIX attribute type. Values are SEC_CHAR, SEC_INT, SEC_LIST, and SEC_BOOL.</td>
</tr>
<tr>
<td>LDAP_Attribute_Name</td>
<td>Specifies the LDAP attribute name.</td>
</tr>
<tr>
<td>LDAP_Value_Type</td>
<td>Specifies the LDAP value type. Values are <code>s</code> for single value and <code>m</code> for multi-value.</td>
</tr>
</tbody>
</table>

### Files

AIX ships 3 sets of attribute mapping files to the `/etc/security/ldap` directory. The first set includes `aixuser.map`, `aixgroup.map`, and `aixid.map`. This set is for use with the AIX specific schema (`aixAccount` and `aixAccessGroup` object classes). The second set includes `2307user.map` and `2307group.map`, and is for use with the `nisSchema` (`posixAccount` and `posixGroup` object classes defined in RFC 2307). The third set includes `aix2307user.map` and `aix2307group.map`, and is for use with `nisSchema` with full AIX support (`posixAccount` and `posixGroup` object classes, plus `aixAuxAccount` and `aixAuxGroup` object classes).

```plaintext
aixuser.map                  Specifies mapping for the `aixAccount` objectclass.
aixgroup.map                Specifies mapping for the `aixAccessGroup` objectclass.
aixid.map                   Specifies mapping for the `aixAdmin` objectclass.
2307user.map                Specifies mapping for the `posixAccount` objectclass.
2307group.map               Specifies mapping for the `posixGroup` objectclass.
aix2307user.map             Specifies mapping for the `posixAccount` and `aixAuxAccount` object-classes.
aix2307group.map            Specifies mapping for the `posixGroup` and `aixAuxGroup` objectclasses.
```

The user and group maps contain an entry that is used to designate the required object class that each user or group must have. This objectclass will be used in the filter for searches performed on user or group entries. As an example, listed below are the default entries for the `keyobjectclass` in the `aix2307user.map` and `aix2307group.map` files. If the LDAP server uses a different object class to provide the required attributes then update the entries accordingly and restart the `secldapclntd` daemon.

```plaintext
aix2307user.map:
    keyobjectclass SEC_CHAR posixgroup s
aix2307group.map:
    keyobjectclass SEC_CHAR posixaccount s
```

440  AIX 5L Version 5.3 Files Reference
The `aixid.map` contains attribute mappings for user and group IDs. The IDs are used when one creates a new LDAP user/group with the `mkuser` or `mkgroup` command.

If an LDAP server uses schema which is not covered by the above 3 sets, you must come up with your own map set. In this case, you must edit the `/etc/security/ldap.cfg` file to configure the client manually.

**Related Information**

The `mksecldap` command and `secldapclntd` daemon.

The `start-secldapclntd`, `stop-secldapclntd`, `restart-secldapclntd`, `ls-secldapclntd` and `flush-secldapclntd` commands.

The `/etc/security/ldap/ldap.cfg` file.

[Lightweight Directory Access Protocol](http://example.com) in *Operating system and device management*.

---

## Locale Definition Source File Format

**Purpose**

Contains one or more categories that describe a locale.

**Description**

A locale definition source file contains one or more categories that describe a locale. Files using this format can be converted into a locale by using the `localedef` command. Locales can be modified only by editing a locale definition source file and then using the `localedef` command again on the new source file.

Locales are not affected by a locale definition source file unless the file is first converted using the `localedef` command.

The locale definition source file sections define categories of locale data. A source file should not contain more than one section for the same category. The following categories are supported:

- **LC_COLLATE**: Defines character or string collation information.
- **LC_CTYPE**: Defines character classification, case conversion, and other character attributes.
- **LC_MESSAGES**: Defines the format for affirmative and negative responses.
- **LC_MONETARY**: Defines rules and symbols for formatting monetary numeric information.
- **LC_NUMERIC**: Defines a list of rules and symbols for formatting non-monetary numeric information.
- **LC_TIME**: Defines a list of rules and symbols for formatting time and date information.

The category definition consists of:

- The category header (category name)
- The associated keyword/value pairs that comprise the category body
- The category trailer (which consists of `END category-name`)

For example:

```plaintext
LC_CTYPE
source for LC_CTYPE category
END LC_CTYPE
```

The source for all of the categories is specified using keywords, strings, character literals, and character symbols. Each keyword identifies either a definition or a rule. The remainder of the statement containing the keyword contains the operands to the keyword. Operands are separated from the keyword by one or
more blank characters. A statement may be continued on the next line by placing a / (slash) as the last character before the new-line character that terminates the line. Lines containing the `comment_char` entry in the first column are treated as comment lines. The default is # (pound sign).

The first category header in the file can be preceded by a line that changes the comment character. It has the following format, starting in column 1:

```
comment_char character
```

where `character` is the new comment character.

Blank lines and lines containing the comment character in the first position are ignored.

A character symbol begins with the < (less-than) character, followed by up to 30 non-control, non-space characters, and ends with the > (greater-than) character. For example, `<A-diaeresis>` is a valid character symbol. Any character symbol referenced in the source file should either be one of the portable character set symbols or should be defined in the provided character set description (charmap) source file.

A character literal is the character itself, or else a decimal, hexadecimal, or octal constant. A decimal constant is of the form:

```
\dxx
```

where `x` is a decimal digit. A hexadecimal constant is of the form:

```
\xd
```

where `d` is a hexadecimal digit. An octal constant is of the form:

```
\d
```

where `d` is an octal digit.

A string is a sequence of character symbols, or literals enclosed by " " (double-quotation marks). For example:

```
"<A-diaeresis> \d65\d120 <B>"
```

The explicit definition of each category in a locale definition source file is not required. When a category is undefined in a locale definition source file, it defaults to the C locale definition.

The first category header in the file can be preceded by a line that changes the escape character used in the file. It has the following format, starting in column 1:

```
escape_char character
```

where `character` is the new escape character.

The escape character defaults to the / (backslash).

**Files**

```
/usr/lib/nls/loc/*
/usr/lib/nls/charmap/*
```

Specifies locale definition source files for supported locales.

Specifies character set description (charmap) source files for supported locales.

**Related Information**

The locale command, localedef command.

Character Set Description (charmap) Source File Format, Locale Method Source File Format.
For specific information about the locale categories and their keywords, see the \texttt{LC\_COLLATE} category, \texttt{LC\_CTYPE} category, \texttt{LC\_MESSAGES} category, \texttt{LC\_MONETARY} category, \texttt{LC\_NUMERIC} category, and \texttt{LC\_TIME} category for the locale definition source file format.

Changing Your Locale in Operating system and device management.

Understanding the Locale Definition Source File in Operating system and device management.

\section*{LC\_COLLATE Category for the Locale Definition Source File Format}

\subsection*{Purpose}
Defines character or string collation information.

\subsection*{Description}
A collation element is the unit of comparison for collation. A collation element may be a character or a sequence of characters. Every collation element in the locale has a set of weights, which determine if the collation element collates before, equal to, or after the other collation elements in the locale. Each collation element is assigned collation weights by the \texttt{localedef} command when the locale definition source file is converted. These collation weights are then used by applications programs that compare strings.

Comparison of strings is performed by comparing the collation weights of each character in the string until either a difference is found or the strings are determined to be equal. This comparison may be performed several times if the locale defines multiple collation orders. For example, in the French locale, the strings are compared using a primary set of collation weights. If they are equal on the basis of this comparison, they are compared again using a secondary set of collation weights. A collating element has a set of collation weights associated with it that is equal to the number of collation orders defined for the locale.

Every character defined in the \texttt{charmap} file (or every character in the portable character set if no \texttt{charmap} file is specified) is itself a collating element. Additional collating elements can be defined using the \texttt{collating-element} statement. The syntax is:

\begin{verbatim}
collating-element character-symbol from string
\end{verbatim}

The \texttt{LC\_COLLATE} category begins with the \texttt{LC\_COLLATE} keyword and ends with the \texttt{END\_LC\_COLLATE} keyword.

The following keywords are recognized in the \texttt{LC\_COLLATE} category:

\begin{verbatim}
copy

collating-element
\end{verbatim}

The \texttt{copy} statement specifies the name of an existing locale to be used as the definition of this category. If a \texttt{copy} statement is included in the file, no other keyword can be specified.

The \texttt{collating-element} statement specifies multicharacter collating elements.

The syntax for the \texttt{collating-element} statement is:

\begin{verbatim}
collating-element <collating-symbol> from <string>
\end{verbatim}

The \texttt{collating-symbol} value defines a collating element that is a string of one or more characters as a single collating element. The \texttt{collating-symbol} value cannot duplicate any symbolic name in the current \texttt{charmap} file, or any other symbolic name defined in this collation definition. The \texttt{string} value specifies a string of two or more characters that define the \texttt{collating-symbol} value. Following are examples of the syntax for the \texttt{collating-element} statement:
collating-element <ch> from <c><h>
collating-element <e-acute> from <acute><e>
collating-element <11> from <1><1>

A collating-symbol value defined by the collating-element statement is recognized only with the LC_COLLATE category.

collating-symbol

The collating-symbol statement specifies collation symbols for use in collation sequence statements.

The syntax for the collating-symbol statement is:

collating-symbol <collating-symbol>

The collating-symbol value cannot duplicate any symbolic name in the current charmap file, or any other symbolic name defined in this collation definition. Following are examples of the syntax for the collating-symbol statement:

collating-symbol <UPPER_CASE>
collating-symbol <HIGH>

A collating-symbol value defined by the collating-symbol statement is recognized only within the LC_COLLATE category.

order_start

The order_start statement must be followed by one or more collation order statements, assigning collation weights to collating elements. This statement is mandatory.

The syntax for the order_start statement is:

order_start <sort-rules>, <sort-rules>,...<sort-rules>
collation order statements
order_end

The <sort-rules> directives have the following syntax:

keyword, keyword,...keyword; keyword, keyword,...keyword

where keyword is one of the keywords forward, backward, and position.

The sort-rules directives are optional. If present, they define the rules to apply during string comparison. The number of specified sort-rules directives defines the number of weights each collating element is assigned (that is, the number of collation orders in the locale). If no sort-rules directives are present, one forward keyword is assumed and comparisons are made on a character basis rather than a string basis. If present, the first sort-rules directive applies when comparing strings using primary weight, the second when comparing strings using the secondary weight, and so on. Each set of sort-rules directives is separated by a ; (semicolon). A sort-rules directive consists of one or more comma-separated keywords. The following keywords are supported:

forward Specifies that collation weight comparisons proceed from the beginning of a string toward the end of the string.
backward Specifies that collation weight comparisons proceed from the end of a string toward the beginning of the string.
position Specifies that collation weight comparisons consider the relative position of elements in the string not subject to the special symbol IGNORE. That is, if strings compare equal, the element with the shortest distance from the starting point of the string collates first.
The **forward** and **backward** keywords are mutually exclusive. Following is an example of the syntax for the `<sort-rules>` directives:

```plaintext
order_start forward; backward, position
```

The optional operands for each collation element are used to define the primary, secondary, or subsequent weights for the collating element. The special symbol `IGNORE` is used to indicate a collating element that is to be ignored when strings are compared.

A collation statement with the `ellipsis` keyword on the left-hand side results in the `collating-element-list` on the right-hand side being applied to every character with an encoding that falls numerically between the character on the left-hand side in the preceding statement and the character on the left-hand side of the following statement. If the `ellipsis` occur in the first statement, it is interpreted as though the preceding line specified the `NUL` character. (The `NUL` character is a character with all bits set to 0.) If the `ellipsis` occur in the last statement, it is interpreted as though the following line specified the greatest encoded value.

An `ellipsis` keyword appearing in place of a `collating-element-list` indicates the weights are to be assigned, for the characters in the identified range, in numerically increasing order from the weight for the character symbol on the left-hand side of the preceding statement.

**Note:** The use of the `ellipsis` keyword results in a locale that may collate differently when compiled with different character set description (`charmap`) source files. For this reason, the `localedef` command issues a warning when the `ellipsis` keyword is encountered.

All characters in the character set must be placed in the collation order, either explicitly or implicitly by using the `UNDEFINED` special symbol. The `UNDEFINED` special symbol includes all coded character set values not specified explicitly or with an ellipsis symbol. These characters are inserted in the character collation order at the point indicated by the `UNDEFINED` special symbol in the order of their character code set values. If no `UNDEFINED` special symbol exists and the collation order does not specify all collation elements from the coded character set, a warning is issued and all undefined characters are placed at the end of the character collation order.

**Examples**

The following is an example of a collation order statement in the `LC_COLLATE` locale definition source file category:

```plaintext
order_start forward;backward
UNDENNED IGNORE;IGNORE
<LOW>  <LOW>;<space>
...  <LOW>;...
<a>  <a>;<a>
<a-acute>  <a>;<a-acute>
<a-grave>  <a>;<a-grave>
<A>  <a>;<A>
<A-acute>  <a>;<A-acute>
<A-grave>  <a>;<A-grave>
<ch>  <ch>;<ch>
<Ch>  <ch>;<Ch>
<s>  <s>;<s>
<s>  <s>;<s>
<eszet>  <eszet>;<eszet>
...  <HIGH>;...
<HIGH>
order_end
```

This example is interpreted as follows:
- The `UNDEFINED` special symbol indicates that all characters not specified in the definition (either explicitly or by the ellipsis symbol) are ignored for collation purposes.
• All collating elements between <space> and <a> have the same primary equivalence class and individual secondary weights based on their coded character set values.
• All characters based on the uppercase or lowercase a character belong to the same primary equivalence class.
• The <<ch>> multicharacter collating element is represented by the <ch> collating symbol and belongs to the same primary equivalence class as the <<ch>> multicharacter collating element.
• The <eszet> character is collated as an s<s> string. That is, one <eszet> character is expanded to two characters before comparing.

Files

/files/lib/nls/loc/*
/files/lib/nls/charmap/*

Specifies locale definition source files for supported locales.
Specifies character set description (charmap) source files for supported locales.

Related Information

The ed, locale, localedef command.

Character Set Description (charmap) Source File Format
Locale Definition Source File Format
Locale Method Source File Format

For specific information about other locale categories and their keywords, see the LC_CTYPE category, LC_MESSAGES category, LC_MONETARY category, LC_NUMERIC category, and LC_TIME category for the locale definition source file format.

Changing Your Locale and Understanding the Locale Definition Source File in Operating system and device management.

LC_CTYPE Category for the Locale Definition Source File Format

Purpose

Defines character classification, case conversion, and other character attributes.

Description

The LC_CTYPE category of a locale definition source file defines character classification, case conversion, and other character attributes. This category begins with an LC_CTYPE category header and terminates with an END LC_CTYPE category trailer.

All operands for LC_CTYPE category statements are defined as lists of characters. Each list consists of one or more semicolon-separated characters or symbolic character names.

The following keywords are recognized in the LC_CTYPE category. In the descriptions, the term automatically included means that an error does not occur if the referenced characters are included or omitted. The characters will be provided if they are missing and will be accepted if they are present.

copy
Specifies the name of an existing locale to be used as the definition of this category. If a copy statement is included in the file, no other keyword can be specified.

upper
Defines uppercase letter characters. No character defined by the cntrl, digit, punct, or space keyword can be specified. At a minimum, the uppercase letters A-Z must be defined.

lower
Defines lowercase letter characters. No character defined by the cntrl, digit, punct, or space keyword can be specified. At a minimum, the lowercase letters a-z must be defined.
alpha Defines all letter characters. No character defined by the \texttt{cntrl}, \texttt{digit}, \texttt{punct}, or \texttt{space} keyword can be specified. Characters defined by the \texttt{upper} and \texttt{lower} keywords are automatically included in this character class.

digit Defines numeric digit characters. Only the digits 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9 can be specified.

alnum Defines alphanumeric characters. No character defined by the \texttt{cntrl}, \texttt{punct}, or \texttt{space} keyword can be specified. Characters defined by the \texttt{alpha} and \texttt{digit} keywords are automatically included in this character class.

space Defines whitespace characters. No character defined by the \texttt{upper}, \texttt{lower}, \texttt{alpha}, \texttt{digit}, \texttt{graph}, \texttt{cntrl}, or \texttt{xdigit} keyword can be specified. At a minimum, the \texttt{<space>}, \texttt{<form-feed>}, \texttt{<newline>}, \texttt{<carriage return>}, \texttt{<tab>}, and \texttt{<vertical-tab>} characters, and any characters defined by the \texttt{blank} keyword, must be specified.

cntrl Defines control characters. No character defined by the \texttt{upper}, \texttt{lower}, \texttt{alpha}, \texttt{digit}, \texttt{punct}, \texttt{graph}, \texttt{print}, \texttt{xdigit}, or \texttt{space} keyword can be specified.

punct Defines punctuation characters. A character defined as the \texttt{<space>} character and characters defined by the \texttt{upper}, \texttt{lower}, \texttt{alpha}, \texttt{digit}, \texttt{cntrl}, or \texttt{xdigit} keyword cannot be specified.

graph Defines printable characters, excluding the \texttt{<space>} character. If this keyword is not specified, characters defined by the \texttt{upper}, \texttt{lower}, \texttt{alpha}, \texttt{digit}, \texttt{xdigit}, and \texttt{punct} keywords are automatically included in this character class. No character defined by the \texttt{cntrl} keyword can be specified.

print Defines printable characters, including the \texttt{<space>} character. If this keyword is not specified, the \texttt{<space>} character and characters defined by the \texttt{upper}, \texttt{lower}, \texttt{alpha}, \texttt{digit}, \texttt{xdigit}, and \texttt{punct} keywords are automatically included in this character class. No character defined by the \texttt{cntrl} keyword can be specified.

xdigit Defines hexadecimal digit characters. The digits 0-9 and the letters A-F and a-f can be specified. The \texttt{xdigit} keyword defaults to its normal class limits.

blank Defines blank characters. If this keyword is not specified, the \texttt{<space>} and \texttt{<horizontal-tab>} characters are included in this character class. Any characters defined by this statement are automatically included in the \texttt{space} keyword class.

charclass Defines one or more locale-specific character class names as strings separated by semicolons. Each named character class can then be defined subsequently in the \texttt{LC\_CTYPE} definition. A character class name consists of at least one, and at most 32 bytes, of alphanumeric characters from the portable character set symbols. The first character of a character class name cannot be a digit. The name cannot match any of the \texttt{LC\_CTYPE} keywords defined in this section.

\texttt{charclass-name} Defines characters to be classified as belonging to the named locale-specific character class. Locale-specific named character classes need not exist in the POSIX locale.

If a class name is defined by a \texttt{charclass} keyword, but no characters are subsequently assigned to it, it represents a class without any characters belonging to it.

The \texttt{charclass-name} can be used as the \texttt{Property} parameter in the \texttt{wctype} subroutine, in regular expressions and shell pattern-matching expressions, and by the \texttt{tolower} command.

toupper Defines the mapping of lowercase characters to uppercase characters. Operands for this keyword consist of semicolon-separated character pairs. Each character pair is enclosed in ( ) (parentheses) and separated from the next pair by a , (comma). The first character in each pair is considered lowercase; the second character is considered uppercase. Only characters defined by the \texttt{lower} and \texttt{upper} keywords can be specified.

tolower Defines the mapping of uppercase characters to lowercase characters. Operands for this keyword consist of semicolon-separated character pairs. Each character pair is enclosed in ( ) (parentheses) and separated from the next pair by a , (comma). The first character in each pair is considered uppercase; the second character is considered lowercase. Only characters defined by the \texttt{lower} and \texttt{upper} keywords can be specified.

The \texttt{tolower} keyword is optional. If this keyword is not specified, the mapping defaults to the reverse mapping of the \texttt{toupper} keyword, if specified. If the \texttt{toupper} and \texttt{tolower} keywords are both unspecified, the mapping for each defaults to that of the C locale.
The **LC_CTYPE** category does not support multicharacter elements. For example, the German sharp-s character is traditionally classified as a lowercase letter. There is no corresponding uppercase letter; in proper capitalization of German text, the sharp-s character is replaced by the two characters `ss`. This kind of conversion is outside of the scope of the `toupper` and `tolower` keywords.

**Examples**

The following is an example of a possible **LC_CTYPE** category listed in a locale definition source file:

```plaintext
LC_CTYPE
# "alpha" is by default "upper" and "lower"
# "alnum" is by default "alpha" and "digit"
# "print" is by default "alnum", "punct" and the space character
# "graph" is by default "alnum" and "punct"
# "tolower" is by default the reverse mapping of "toupper"

upper  <A>;<B>;<C>;<D>;<E>;<F>;<G>;<H>;<I>;<J>;<K>;<L>;<M>;<N>;<O>;<P>;<Q>;<R>;<S>;<T>;<U>;<V>;<W>;<X>;<Y>;<Z>
lower  <a>;<b>;<c>;<d>;<e>;<f>;<g>;<h>;<i>;<j>;<k>;<l>;<m>;<n>;<o>;<p>;<q>;<r>;<s>;<t>;<u>;<v>;<w>;<x>;<y>;<z>

digit   <zero>;<one>;<two>;<three>;<four>;<five>;<six>;<seven>;<eight>;<nine>

space   <tab>;<newline>;<vertical-tab>;<form-feed>;<carriage-return>;<space>

cntrl <alert>;<backspace>;<newline>;<vertical-tab>;<form-feed>;<carriage-return>;<NUL>;<SOH>;<STX>;<ETX>;<EOT>;<ENQ>;<ACK>;<SO>;<SI>;<DLE>;<DC1>;<DC2>;<DC3>;<DC4>;<N AK>;<SYN>;<ETB>;<CAN>;<SUB>;<ESC>;<IS4>;<IS3>;<IS2>;<IS1>;<DEL>

punct <exclamation-mark>;<quotation-mark>;<number-sign>;<dollar-sign>;<percent-sign>;<ampersand>;<asterisk>;<apostrophe>;<left-parenthesis>;<right-parenthesis>;<plus-sign>;<comma>;<hyphen>;<period>;<slash>;<colon>;<semicolon>;<less-than-sign>;<equals-sign>;<greater-than-sign>;<question-mark>;<commercial-at>;<left-square-bracket>;<right-square-bracket>;<vertical-line>;<tilde>;<right-curly-bracket>

xdigit <zero>;<one>;<two>;<three>;<four>;<five>;<six>;<seven>;<eight>;<nine>;<A>;<B>;<C>;<D>;<E>;<F>;<a>;<b>;<c>;<d>;<e>;<f>

blank  <space>;<tab>

toupper ( <a>,<A> ); ( <b>,<B> ); ( <c>,<C> ); ( <d>,<D> ); ( <e>,<E> ); ( <f>,<F> ); ( <g>,<G> ); ( <h>,<H> ); ( <i>,<I> ); ( <j>,<J> ); ( <k>,<K> ); ( <l>,<L> ); ( <m>,<M> ); ( <n>,<N> ); ( <o>,<O> ); ( <p>,<P> ); ( <q>,<Q> ); ( <r>,<R> ); ( <s>,<S> ); ( <t>,<T> ); ( <u>,<U> ); ( <v>,<V> ); ( <w>,<W> ); ( <x>,<X> ); ( <y>,<Y> ); ( <z>,<Z> )

END LC_CTYPE
```

**Files**

```
/usr/lib/nls/loc/*
/usr/lib/nls/charmap/*
```

Specifies locale definition source files for supported locales.

Specifies character set description (charmap) source files for supported locales.
The `locale` command, `localedef` command, and `tr` command.

The `wctype` subroutine.


For specific information about other locale categories and their keywords, see the `LC_COLLATE` category, `LC_MESSAGES` category, `LC_MONETARY` category, `LC_NUMERIC` category, and `LC_TIME` category for the locale definition source file format.

Changing Your Locale and Understanding the Locale Definition Source File in Operating system and device management.

**LC_MESSAGES Category for the Locale Definition Source File Format**

**Purpose**

Defines the format for affirmative and negative system responses.

**Description**

The `LC_MESSAGES` category of a locale definition source file defines the format for affirmative and negative system responses. This category begins with an `LC_MESSAGES` category header and terminates with an `END LC_MESSAGES` category trailer.

All operands for the `LC_MESSAGES` category are defined as strings or extended regular expressions enclosed by " " (double-quotation marks). These operands are separated from the keyword they define by one or more blanks. Two adjacent " " (double-quotation marks) indicate an undefined value. The following keywords are recognized in the `LC_MESSAGES` category:

- **copy**
  Specifies the name of an existing locale to be used as the definition of this category. If a `copy` statement is included in the file, no other keyword can be specified.

- **yesexpr**
  Specifies an extended regular expression that describes the acceptable affirmative response to a question expecting an affirmative or negative response.

- **noexpr**
  Specifies an extended regular expression that describes the acceptable negative response to a question expecting an affirmative or negative response.

- **yesstr**
  A colon-separated string of acceptable affirmative responses. This string is accessible to applications through the `nl_langinfo` subroutine as `nl_langinfo(YESSTR)`.

- **nostr**
  A colon-separated string of acceptable negative responses. This string is accessible to applications through the `nl_langinfo` subroutine as `nl_langinfo(NOSTR)`.

**Examples**

The following is an example of a possible `LC_MESSAGES` category listed in a locale definition source file:

```plaintext
LC_MESSAGES
#
yesexpr "([yY][[:alpha:]]+)([OK])"
noexpr "([nN][[:alpha:]]+)"
yesstr "Y:y:yes"
nostr "N:n:no"
#
END LC_MESSAGES
```
Files

/specs/locale/loc/*
/specs/locale/charmap/*

Specifies locale definition source files for supported locales.
Specifies character set description (charmap) source files for supported locales.

Related Information

The `locale` command, `localedef` command.


For specific information about other locale categories and their keywords, see the `LC_COLLATE` category, `LC_CTYPE` category, `LC_MONETARY` category, `LC_NUMERIC` category, and `LC_TIME` category for the locale definition source file format.

Changing Your Locale and Understanding the Locale Definition Source File in Operating system and device management.

---

**LC_MONETARY Category for the Locale Definition Source File Format**

**Purpose**

Defines rules and symbols for formatting monetary numeric information.

**Description**

The `LC_MONETARY` category of a locale definition source file defines rules and symbols for formatting monetary numeric information. This category begins with an `LC_MONETARY` category header and terminates with an `END LC_MONETARY` category trailer.

All operands for the `LC_MONETARY` category keywords are defined as string or integer values. String values are enclosed by " " (double-quotation marks). All values are separated from the keyword they define by one or more spaces. Two adjacent double-quotation marks indicate an undefined string value. A -1 indicates an undefined integer value. The following keywords are recognized in the `LC_MONETARY` category:

- `copy` Specifies the name of an existing locale to be used as the definition of this category. If a `copy` statement is included in the file, no other keyword can be specified.
- `int_curr_symbol` Specifies the string used for the international currency symbol. The operand for the `int_curr_symbol` keyword is a four-character string. The first three characters contain the alphabetic international-currency symbol. The fourth character specifies a character separator between the international currency symbol and a monetary quantity.
- `currency_symbol` Specifies the string used for the local currency symbol.
- `mon_decimal_point` Specifies the string used for the decimal delimiter used to format monetary quantities.
- `mon_thousands_sep` Specifies the character separator used for grouping digits to the left of the decimal delimiter in formatted monetary quantities.
**mon_grouping**

Specifies a string that defines the size of each group of digits in formatted monetary quantities. The operand for the mon_grouping keyword consists of a sequence of semicolon-separated integers. Each integer specifies the number of digits in a group. The initial integer defines the size of the group immediately to the left of the decimal delimiter. The following integers define succeeding groups to the left of the previous group. If the last integer is not -1, the size of the previous group (if any) is repeatedly used for the remainder of the digits. If the last integer is -1, no further grouping is performed.

The following is an example of the interpretation of the mon_grouping statement. Assuming the value to be formatted is 123456789 and the operand for the mon_thousands_sep keyword is ' (single-quotation mark), the following results occur:

<table>
<thead>
<tr>
<th>mon_grouping Value</th>
<th>Formatted Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>3; -1</td>
<td>123456’789</td>
</tr>
<tr>
<td>3</td>
<td>123’456’789</td>
</tr>
<tr>
<td>3; 2; -1</td>
<td>1234’56’789</td>
</tr>
<tr>
<td>3; 2</td>
<td>12’34’56’789</td>
</tr>
<tr>
<td>-1</td>
<td>123456789</td>
</tr>
</tbody>
</table>

**positive_sign**

Specifies the string used to indicate a nonnegative-valued formatted monetary quantity.

**negative_sign**

Specifies the string used to indicate a negative-valued formatted monetary quantity.

**int_frac_digits**

Specifies an integer value representing the number of fractional digits (those after the decimal delimiter) to be displayed in a formatted monetary quantity using the int_curr_symbol value.

**frac_digits**

Specifies an integer value representing the number of fractional digits (those after the decimal delimiter) to be displayed in a formatted monetary quantity using the currency_symbol value.

**p_cs_precedes**

Specifies an integer value indicating whether the int_curr_symbol or currency_symbol string precedes or follows the value for a nonnegative formatted monetary quantity. The following integer values are recognized:

- 0: Indicates that the currency symbol follows the monetary quantity.
- 1: Indicates that the currency symbol precedes the monetary quantity.

**p_sep_by_space**

Specifies an integer value indicating whether the int_curr_symbol or currency_symbol string is separated by a space from a nonnegative formatted monetary quantity. The following integer values are recognized:

- 0: Indicates that no space separates the currency symbol from the monetary quantity.
- 1: Indicates that a space separates the currency symbol from the monetary quantity.
- 2: Indicates that a space separates the currency symbol and the positive_sign string, if adjacent.

**n_cs_precedes**

Specifies an integer value indicating whether the int_curr_symbol or currency_symbol string precedes or follows the value for a negative formatted monetary quantity. The following integer values are recognized:

- 0: Indicates that the currency symbol follows the monetary quantity.
- 1: Indicates that the currency symbol precedes the monetary quantity.
mon_grouping Value
n_sep_by_space

Specifies an integer value indicating whether the int_curr_symbol or currency_symbol string is separated by a space from a negative formatted monetary quantity. The following integer values are recognized:

0  Indicates that no space separates the currency symbol from the monetary quantity.
1  Indicates that a space separates the currency symbol from the monetary quantity.
2  Indicates that a space separates the currency symbol and the negative_sign string, if adjacent.

p_sign_posn

Specifies an integer value indicating the positioning of the positive_sign string for a nonnegative formatted monetary quantity. The following integer values are recognized:

0  Indicates that a left_parenthesis and right_parenthesis symbol enclose both the monetary quantity and the int_curr_symbol or currency_symbol string.
1  Indicates that the positive_sign string precedes the quantity and the int_curr_symbol or currency_symbol string.
2  Indicates that the positive_sign string follows the quantity and the int_curr_symbol or currency_symbol string.
3  Indicates that the positive_sign string immediately precedes the int_curr_symbol or currency_symbol string.
4  Indicates that the positive_sign string immediately follows the int_curr_symbol or currency_symbol string.

n_sign_posn

Specifies an integer value indicating the positioning of the negative_sign string for a negative formatted monetary quantity. The following integer values are recognized:

0  Indicates that a left_parenthesis and right_parenthesis symbol enclose both the monetary quantity and the int_curr_symbol or currency_symbol string.
1  Indicates that the negative_sign string precedes the quantity and the int_curr_symbol or currency_symbol string.
2  Indicates that the negative_sign string follows the quantity and the int_curr_symbol or currency_symbol string.
3  Indicates that the negative_sign string immediately precedes the int_curr_symbol or currency_symbol string.
4  Indicates that the negative_sign string immediately follows the int_curr_symbol or currency_symbol string.

debit_sign

Specifies the string used for the debit symbol (DB) to indicate a nonnegative formatted monetary quantity.

credit_sign

Specifies the string used for the credit symbol (CR) to indicate a negative formatted monetary quantity.

left_parenthesis

Specifies the character, equivalent to a ( (left parenthesis), used by the p_sign_posn and n_sign_posn statements to enclose a monetary quantity and currency symbol.

right_parenthesis

Specifies the character, equivalent to a ) (right parenthesis), used by the p_sign_posn and n_sign_posn statements to enclose a monetary quantity and currency symbol.

A unique customized monetary format can be produced by changing the value of a single statement. For example, the following table shows the results of using all combinations of defined values for the p_cs_precedes, p_sep_by_space, and p_sign_posn statements.
### Table 5. Results of Various Locale Variable Value Combinations

<table>
<thead>
<tr>
<th>p_cs_precedes</th>
<th>p_sign_posn</th>
<th>p_sep_by_space =</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_cs_precedes</td>
<td>p_sign_posn</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>2 ($1.25)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 ($1.25)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 ($1.25)</td>
</tr>
<tr>
<td>p_sign_posn</td>
<td>1</td>
<td>2 + $1.25</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 +$ 1.25</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 +$1.25</td>
</tr>
<tr>
<td>p_sign_posn</td>
<td>2</td>
<td>2 $1.25 +</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 $ 1.25+</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 $1.25+</td>
</tr>
<tr>
<td>p_sign_posn</td>
<td>3</td>
<td>2 + $1.25</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 +$ 1.25</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 +$1.25</td>
</tr>
<tr>
<td>p_sign_posn</td>
<td>4</td>
<td>2 $ +1.25</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 $+ 1.25</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 $+1.25</td>
</tr>
<tr>
<td>p_cs_precedes</td>
<td>p_sign_posn</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>2 (1.25 $)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 (1.25 $)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 (1.25$)</td>
</tr>
<tr>
<td>p_sign_posn</td>
<td>1</td>
<td>2 +1.25 $</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 +1.25 $</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 +1.25$</td>
</tr>
<tr>
<td>p_sign_posn</td>
<td>2</td>
<td>2 1.25$ +</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 1.25 $+</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 1.25$+</td>
</tr>
<tr>
<td>p_sign_posn</td>
<td>3</td>
<td>2 1.25+ $</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 1.25 +$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 1.25+$</td>
</tr>
<tr>
<td>p_sign_posn</td>
<td>4</td>
<td>2 1.25$ +</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 1.25 $+</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 1.25$+</td>
</tr>
</tbody>
</table>

**Example**

The following is an example of a possible `LC_MONETARY` category listed in a locale definition source file:

```bash
LC_MONETARY
#
int_curr_symbol "<$><$><$>"
```
Files

/usr/lib/nls/loc/* Specifies locale definition source files for supported locales.
/usr/lib/nls/charmap/* Specifies character set description (charmap) source files for supported locales.

Related Information
The locale command, localedef command.


For specific information about other locale categories and their keywords, see the LC_COLLATE category, LC_CTYPE category, LC_MESSAGES category, LC_NUMERIC category, and LC_TIME category for the locale definition source file format.

Changing Your Locale and Understanding the Locale Definition Source File in Operating system and device management.

LC_NUMERIC Category for the Locale Definition Source File Format

Purpose
Defines rules and symbols for formatting non-monetary numeric information.

Description
The LC_NUMERIC category of a locale definition source file defines rules and symbols for formatting non-monetary numeric information. This category begins with an LC_NUMERIC category header and terminates with an END LC_NUMERIC category trailer.

All operands for the LC_NUMERIC category keywords are defined as string or integer values. String values are enclosed by " " (double-quotation marks). All values are separated from the keyword they define by one or more spaces. Two adjacent double-quotation marks indicate an undefined string value. A -1 indicates an undefined integer value. The following keywords are recognized in the LC_NUMERIC category:
**copy**

The *copy* statement specifies the name of an existing locale to be used as the definition of this category. If a *copy* statement is included in the file, no other keyword can be specified.

**decimal_point**

Specifies the string used for the decimal delimiter used to format numeric, non-monetary quantities.

**thousands_sep**

Specifies the string separator used for grouping digits to the left of the decimal delimiter in formatted numeric, non-monetary quantities.

**grouping**

Defines the size of each group of digits in formatted monetary quantities. The operand for the *grouping* keyword consists of a sequence of semicolon-separated integers. Each integer specifies the number of digits in a group. The initial integer defines the size of the group immediately to the left of the decimal delimiter. The following integers define succeeding groups to the left of the previous group. If the last integer is not -1, the size of the previous group (if any) is used repeatedly for the remainder of the digits. If the last integer is -1, no further grouping is performed.

The following is an example of the interpretation of the *grouping* statement. Assuming the value to be formatted is 123456789 and the operand for the *thousands_sep* keyword is ‘`’ (single quotation mark) the following results occur:

<table>
<thead>
<tr>
<th>Grouping Value</th>
<th>Formatted Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>3;1-1</td>
<td>123456’789</td>
</tr>
<tr>
<td>3</td>
<td>123’456’789</td>
</tr>
<tr>
<td>3;2;1-1</td>
<td>1234’56’789</td>
</tr>
<tr>
<td>3;2</td>
<td>12’34’56’789</td>
</tr>
<tr>
<td>-1</td>
<td>123456789</td>
</tr>
</tbody>
</table>

**Examples**

Following is an example of a possible *LC_NUMERIC* category listed in a locale definition source file:

```
LC_NUMERIC
    #
decimal_point     "<period>"
    thousands_sep     "<comma>"
    grouping          <3>
    #
END LC_NUMERIC
```

**Files**

```
/usr/lib/nls/loc/* specifies locale definition source files for supported locales.
/usr/lib/nls/charmap/* specifies character set description (charmap) source files for supported locales.
```

**Related Information**

The following commands are useful:

- *locale*
- *localedef*

For specific information about other locale categories and their keywords, see the *LC_COLLATE* category, *LC_CTYPE* category, *LC_MESSAGES* category, *LC_MONETARY* category, and *LC_TIME* category for the locale definition source file format.

Changing Your Locale and Understanding the Locale Definition Source File in *Operating system and device management*.
LC_TIME Category for the Locale Definition Source File Format

Purpose
Defines rules and symbols for formatting time and date information.

Description
The LC_TIME category of a locale definition source file defines rules and symbols for formatting time and date information. This category begins with an LC_TIME category header and terminates with an END LC_TIME category trailer.

Keywords
All operands for the LC_TIME category keywords are defined as string or integer values. String values are enclosed by " " (double-quotation marks). All values are separated from the keyword they define by one or more spaces. Two adjacent double-quotation marks indicate an undefined string value. A -1 indicates an undefined integer value. Field descriptors are used by commands and subroutines that query the LC_TIME category to represent elements of time and date formats. The following keywords are recognized in the LC_TIME category:

copy
The copy statement specifies the name of an existing locale to be used as the definition of this category. If a copy statement is included in the file, no other keyword can be specified.

abday
Defines the abbreviated weekday names corresponding to the %a field descriptor. Recognized values consist of 7 semicolon-separated strings. Each string must be of equal length and contain 5 characters or less. The first string corresponds to the abbreviated name (Sun) for the first day of the week (Sunday), the second to the abbreviated name for the second day of the week, and so on.

day
Defines the full spelling of the weekday names corresponding to the %A field descriptor. Recognized values consist of seven semicolon-separated strings. The first string corresponds to the full spelling of the name of the first day of the week (Sunday), the second to the name of the second day of the week, and so on.

abmon
Defines the abbreviated month names corresponding to the %b field descriptor. Recognized values consist of 12 semicolon-separated strings. Each string must be of equal length and contain 5 characters or less. The first string corresponds to the abbreviated name (Jan) for the first month of the year (January), the second to the abbreviated name for the second month of the year, and so on.

mon
Defines the full spelling of the month names corresponding to the %B field descriptor. Recognized values consist of 12 semicolon-separated strings. The first string corresponds to the full spelling of the name of the first month of the year (January), the second to the full spelling of the name for the second month of the year, and so on.

d_t_fmt
Defines the string used for the standard date and time format corresponding to the %c field descriptor. The string can contain any combination of characters and field descriptors.

d_fmt
Defines the string used for the standard date format corresponding to the %x field descriptor. The string can contain any combination of characters and field descriptors.

t_fmt
Defines the string used for the standard time format corresponding to the %X field descriptor. The string can contain any combination of characters and field descriptors.

am_pm
Defines the strings used to represent ante meridiem (before noon) and post meridiem (after noon) corresponding to the %p field descriptor. Recognized values consist of two semicolon-separated strings. The first string corresponds to the ante meridiem designation, the last string to the post meridiem designation.

t_fmt_ampm
Defines the string used for the standard 12-hour time format that includes an am_pm value (the %p field descriptor). This statement corresponds to the %r field descriptor. The string can contain any combination of characters and field descriptors.
era  Defines how the years are counted and displayed for each era (or emperor’s reign) in a locale, corresponding to the %E field descriptor modifier. For each era, there must be one string in the following format:

direction:offset:start_date:end_date:name:format

The variables for the era-string format are defined as follows:

direction  Specifies a - (minus sign) or + (plus sign) character. The plus sign character indicates that years count in the positive direction when moving from the start date to the end date. The minus sign character indicates that years count in the negative direction when moving from the start date to the end date.

offset   Specifies a number representing the first year of the era.

start_date  Specifies the starting date of the era in the yyyy/mm/dd format, where yyyy, mm, and dd are the year, month, and day, respectively. Years prior to the year AD 1 are represented as negative numbers. For example, an era beginning March 5th in the year 100 BC would be represented as -100/03/05.

date   Specifies the ending date of the era in the same form used for the start_date variable or one of the two special values -* or +*. A -* value indicates that the ending date of the era extends backward to the beginning of time. A +* value indicates that the ending date of the era extends forward to the end of time. Therefore, the ending date can be chronologically before or after the starting date of the era. For example, the strings for the Christian eras AD and BC would be entered as follows:

-0000/01/01:+*:AD:%o
-0001/12/31:-*:BC:%o

name   Specifies a string representing the name of the era that is substituted for the %N field descriptor.

format  Specifies a string for formatting the %E field descriptor. This string is usually a function of the %o and %N field descriptors.

An era value consists of one string for each era. If more than one era is specified, each era string is separated by a ; (semicolon).

era_year  Defines the string used to represent the year in alternate-era format corresponding to the %Ey field descriptor. The string can contain any combination of characters and field descriptors.

era_d_fmt  Defines the string used to represent the date in alternate-era format corresponding to the %Ex field descriptor. The string can contain any combination of characters and field descriptors.

era_t_fmt  Defines the alternative time format of the locale, as represented by the %EX field descriptor for the strftime subroutine.

era_d_t_fmt  Defines the alternative date and time format of the locale, as represented by the %Ec field descriptor for the strftime subroutine.

alt_digits  Defines alternate strings for digits corresponding to the %o field descriptor. Recognized values consist of a group of semicolon-separated strings. The first string represents the alternate string for 0, the second string represents the alternate string for one, and so on. A maximum of 100 alternate strings can be specified.

Field Descriptors

The LC_TIME locale definition source file uses field descriptors to represent elements of time and date formats. Combinations of these field descriptors create other field descriptors or create time-and-date format strings. When used in format strings containing field descriptors and other characters, field descriptors are replaced by their current values. All other characters are copied without change. The following field descriptors are used by commands and subroutines that query the LC_TIME category for time formatting:
%a Represents the abbreviated weekday name (for example, Sun) defined by the abday statement.
%A Represents the full weekday name (for example, Sunday) defined by the day statement.
%b Represents the abbreviated month name (for example, Jan) defined by the abmon statement.
%B Represents the full month name (for example, January) defined by the month statement.
%c Represents the time-and-date format defined by the d_t_fmt statement.
%A Represents the century as a decimal number (00 to 99).
%d Represents the day of the month as a decimal number (01 to 31).
%D Represents the date in %m/%d/%y format (for example, 01/31/91).
%e Represents the day of the month as a two-digit field. If the day of the month is not a two-digit number, the leading digit is filled with a space character.
%Ec Specifies the locale’s alternate appropriate date and time representation.
%EC Specifies the name of the base year (period) in the locale’s alternate representation.
%Ex Specifies the locale’s alternate date representation.
%EX Specifies the locale’s alternate time representation.
%Ey Specifies the offset from the %EC (year only) field descriptor in the locale’s alternate representation.
%EY Specifies the full alternate year representation.
%Od Specifies the day of the month using the locale’s alternate numeric symbols.
%Oe Specifies the day of the month using the locale’s alternate numeric symbols.
%OH Specifies the hour (24-hour clock) using the locale’s alternate numeric symbols.
%OI Specifies the hour (12-hour clock) using the locale’s alternate numeric symbols.
%Om Specifies the month using the locale’s alternate numeric symbols.
%OM Specifies the minutes using the locale’s alternate numeric symbols.
%OS Specifies the seconds using the locale’s alternate numeric symbols.
%OU Specifies the week number of the year (Sunday as the first day of the week) using the locale’s alternate numeric symbols.
%OW Specifies the weekday as a number in the locale’s alternate representation (Sunday = 0).
%Oy Specifies the year (offset from the %C field descriptor) in alternate representation.
%h Represents the abbreviated month name (for example, Jan) defined by the abmon statement. This field descriptor is a synonym for the %b field descriptor.
%H Represents the 24-hour clock hour as a decimal number (00 to 23).
%I Represents the 12-hour clock hour as a decimal number (01 to 12).
%j Represents the day of the year as a decimal number (001 to 366).
%m Represents the month of the year as a decimal number (01 to 12).
%M Represents the minutes of the hour as a decimal number (00 to 59).
%n Specifies a new-line character.
%N Represents the alternate era name.
%o Represents the alternate era year.
%p Represents the a.m. or p.m. string defined by the am_pm statement.
%r Represents the 12-hour clock time with a.m./p.m. notation as defined by the t_fmt_ampm statement.
%S Represents the seconds of the minute as a decimal number (00 to 59).
%t Specifies a tab character.
%T Represents 24-hour clock time in the format %H:%M:%S (for example, 16:55:15).
%U Represents the week of the year as a decimal number (00 to 53). Sunday, or its equivalent as defined by the day statement, is considered the first day of the week for calculating the value of this field descriptor.
%w Represents the day of the week as a decimal number (0 to 6). Sunday, or its equivalent as defined by the day statement, is considered as 0 for calculating the value of this field descriptor.
%W Represents the week of the year as a decimal number (00 to 53). Monday, or its equivalent as defined by the day statement, is considered the first day of the week for calculating the value of this field descriptor.
%x Represents the date format defined by the d_fmt statement.
%X Represents the time format defined by the t_fmt statement.
%y Represents the year of the century (00 to 99).
Note: When the environment variable XPG_TIME_FMT=ON, %y is the year within the century.
When a century is not otherwise specified, values in the range 69-99 refer to years in the twentieth century (1969 to 1999, inclusive); values in the range 00-68 refer to 2000 to 2068, inclusive.
%Y Represents the year as a decimal number (for example, 1989).
%Z Represents the time-zone name, if one can be determined (for example, EST); no characters are displayed if a time zone cannot be determined.
%% Specifies a % (percent sign) character.

Example
The following is an example of a possible LC_TIME category listed in a locale definition source file:

```
LC_TIME
#
# Abbreviated weekday names (%a)
abday  "<S><u><n>";"<M><o><n>";"<T><u><e>";"<W><e><d>";"<T><h><u>";"<F><r><i>";"<S><a><t>"
    "<S><q><a><t><a><t>"
#
# Full weekday names (%A)
day    "<S><u><n><d><a><y>";"<M><o><n><d><a><y>";"<T><u><e><s><d><a><y>";"<W><e><d><n><e><s><d><a><y>
          "<T><h><u><r><s><d><a><y>";"<F><r><i><d><a><y>";"<S><a><t><u><r><d><a><y>
    "<S><q><a><t><a><t><a><t>"
#
# Abbreviated month names (%b)
abmon  "<J><a><n>";"<F><e><b>";"<M><a><r>";"<A><p><r>";"<M><a><y>";"<J><u><n>";"<J><u><l>";"<A><u><g>"
          "<S><e><p>";"<O><c><t>";"<N><o><v>";"<D><e><c>"
#
# Full month names (%B)
mon    "<J><a><n><u><a><r><y>";"<F><e><b><r><u><a><r><y>";"<M><a><r><c><h>";"<A><p><r><i><l>"
          "<M><a><y>";"<J><u><n>";"<J><u><l><y>";"<A><u><g><u><s><t>"
          "<S><e><p><t><e><m><b><e><r>";"<O><c><t><o><b><e><r>";"<N><o><v><e><m><b><e><r>"
    "<D><e><c><e><m><b><e><r>"
#
# Date and time format (%c)
d_t_fmt  "%a %b %d %H:%M:%S %Y"
#
# Date format (%x)
d_fmt    "%m/%d/%y"
#
# Time format (%X)
t_fmt    "%H:%M:%S"
#
# Equivalent of AM/PM (%p)
am_pm    "<A><M>";"<P><M>"
#
# 12-hour time format (%r)
t_fmt_ampm  "%I:%M:%S %p"
#
# Era (%n)
era      "+0:0000/01/01:+*:AD:%o %N";
    "+1:-0001/12/31:-*:BC:%o %N"
-era_year  ""
-era_d_fmt  ""
alt_digits  "<0><t><h>";"<1><t><h><t><h>";"<2><n><t><h><d>";"<3><r><d><d>"
            "<4><t><h><t><h>";"<5><t><h><t><h>";"<6><t><h><t><h>";"<7><t><h><t><h>
            "<8><t><h><t><h>";"<9><t><h><t><h>";"<1><t><h><t><h>"
#
END LC_TIME
```
Files

/specs/locale/*
/specs/charmap/*

Specifies locale definition source files for supported locales.
Specifies character set description (charmap) source files for supported locales.

Related Information

The `locale` command, `localedef` command.

The `strftime` subroutine.


For specific information about other locale categories and their keywords, see the `LC_COLLATE` category, `LC_CTYPE` category, `LC_MESSAGES` category, `LC_MONETARY` category, and `LC_NUMERIC` category for the locale definition source file format.

Changing Your Locale and Understanding the Locale Definition Source File in Operating system and device management.

Locale Method Source File Format

Purpose

Specifies the methods to be overridden when constructing a locale.

Description

The `methods` source file maps methods names to the National Language Support (NLS) subroutines that implement those methods. The `methods` file also specifies the libraries where the implementing subroutines are stored.

The methods correspond to those subroutines that require direct access to the data structures representing locale data.

The following is the expected grammar for a `methods` file:

```
method_def : "METHODS"
  | method_assign_list "END METHODS"
  ;
method_assign_list :
  method_assign_list method_assign
  | method_assign_list
  | method_assign
  ;
method_assign :
  "csid" meth_name meth_lib_path
  | "fnmatch" meth_name meth_lib_path
  | "get_wctype" meth_name meth_lib_path
  | "is_wctype" meth_name meth_lib_path
  | "mblen" meth_name meth_lib_path
  | "mbstombs" meth_name meth_lib_path
  | "mbtowcs" meth_name meth_lib_path
  | "mbtowc" meth_name meth_lib_path
  | "mbstomp" meth_name meth_lib_path
  | "pctombs" meth_name meth_lib_path
  | "pctomb" meth_name meth_lib_path
```
meth_name: global_name
  | cfunc_name

global_name:
  "CSID_STD"
  "FNMATCH_C"
  "FNMATCH_STD"
  "GET_WCTYPE_STD"
  "IS_WCTYPE_SB"
  "IS_WCTYPE_STD"
  "LOCALECONV_STD"
  "MBLEN_932"
  "MBLEN_EUCJP"
  "MBLEN_SB"
  "__MBTOPCS_932"
  "__MBTOPCS_EUCJP"
  "__MBTOPCS_SB"
  "MBSTOWCS_932"
  "MBSTOWCS_EUCJP"
  "MBSTOWCS_SB"
  "__MBTOPC_932"
  "__MBTOPC_EUCJP"
  "__MBTOPC_SB"
  "MBTOWC_932"
  "MBTOWC_EUCJP"
  "MBTOWC_SB"
  "NL_MONINFO"
  "NL_NUMINFO"
  "NL_RESPINFO"
  "NL_TIMINFO"
  "__PCSTOMBS_932"
  "__PCSTOMBS_EUCJP"
  "__PCSTOMBS_SB"
  "__PCTOMB_932"
  "__PCTOMB_EUCJP"
  "__PCTOMB_SB"
  "RECOMP_STD"
  "REGERROR_STD"
  "REGEXEC_STD"
  "REGFREE_STD"
  "RPMATCH_C"
  "RPMATCH_STD"
  "STRC Coll_C"
  "STRC Coll_SB"
  "STRC Coll_STD"
  "STRFMON_STD"
"STRFTIME_STD"
"STRPTIME_STD"
"STRXFRM_C"
"STRXFRM_SB"
"STRXFRM_STD"
"TOWLOWER_STD"
"TOWUPPER_STD"
"WCSCOLL_C"
"WCSCOLL_STD"
"WCSFTIME_STD"
"WCSID_STD"
"WCSTOMBS_932"
"WCSTOMBS_EUCJP"
"WCSTOMBS_SB"
"WCSWIDTH_932"
"WCSWIDTH_EUCJP"
"WCSWIDTH_LATIN"
"WCSXFRM_C"
"WCSXFRM_STD"
"WCTOMB_932"
"WCTOMB_EUCJP"
"WCTOMB_SB"
"WCWIDTH_932"
"WCWIDTH_EUCJP"
"WCWIDTH_LATIN"

Where **cfunc_name** is the name of a user supplied subroutine, and **meth_lib_path** is an optional path name for the library containing the specified subroutine.

**Note:** If a 64-bit locale is to be created, then **meth_lib_path** must specify the path for a single archive for the two shared objects, one 32-bit and the other 64-bit, containing the specified subroutines. Specifying separate paths to the 32-bit and 64-bit shared objects will result in **localedef** failing due to incompatible **XCOFF** format.

The **localedef** command parses this information to determine the methods to be used for this locale. The following subroutines must be specified in the **method** file:

- __mbtopc
- __mbstopcs
- __pctomb
- __pcstombs
- mblen
- mbstowcs
- mbtowc
- wcstombs
- wcswidth
- wctomb
- wcwidth

Any other method not specified in the **method** file retains the default.

Mixing of **cfunc_name** values and **global_name** values is not allowed. A **method** file should not include both. If the **localedef** command receives a **method** file containing both **cfunc_name** values and **global_name** values, an error is generated and the locale is not created.
It is not mandatory that the **METHODS** section specify the library name. If an individual method does not specify a library, the method inherits the most recently specified library. The `libc.a` library is the default library.

The method for the `mbtowc` and `wcwidth` subroutines should avoid calling other methods where possible.

An understanding of how the `__mbtopc`, `__mbstopcs`, `__pctomb`, and `__pcstombs` subroutines process wide characters is useful when constructing a method file. These subroutines should not be used in applications programs.

**__mbtopc Subroutine**

The `__mbtopc` subroutine converts a character to a process code.

The syntax for the `__mbtopc` subroutine is as follows:

```c
size_t __mbtopc(PC, S, LenS, Err)
wchar_t * PC;
uchar * S;
size_t LenS;
int * Err;
```

The input buffer pointed to by the `S` parameter contains the number of bytes of character data specified in the `LenS` parameter. The `__mbtopc` subroutine attempts to convert the character to a process code. If a valid character is found in the input buffer pointed to by the `S` parameter, the character is converted and stored in the `PC` parameter, and the number of bytes in the character is returned.

If the number of bytes specified by the `LenS` parameter in the input buffer pointed to by the `S` parameter form an invalid character, the subroutine returns 0 and sets the `Err` parameter to the value -1. If a character cannot be formed in the number of bytes specified by the `LenS` parameter or less, the subroutine returns 0 and sets the `Err` parameter to the number of bytes required to form a character beginning with the data pointed to by the `S` parameter.

The parameters have the following values:

- `PC` Points to a wide character to contain the converted character.
- `S` Points to the buffer of character data to be converted.
- `LenS` Specifies the number of bytes of character data pointed to by the `S` parameter.
- `Err` Specifies an error value indicating why the conversion failed.

**__mbstopcs Subroutine**

The `__mbstopcs` subroutine converts a character string to a process code string.

The syntax for the `__mbstopcs` subroutine is as follows:

```c
size_t __mbstopcs(PC, LenPC, S, LenS, StopCh, EndPtr, Err)
wchar_t * PC;
size_t LenPC;
uchar * S;
size_t LenS;
uchar StopCh;
uchar ** EndPtr;
int * Err;
```

The input buffer pointed to by the `S` parameter contains the number of bytes of character data specified in the `LenS` parameter. The `__mbstopcs` subroutine attempts to convert the character data to process codes. The conversion of characters continues until one of the following occurs:
• The number of bytes specified by the LenS parameter have been converted.
• The number of characters specified by the LenPC parameter have been converted.
• The byte value specified in the StopCh parameter is encountered in the input buffer pointed to by the S parameter.
• An invalid or incomplete character is found in the input buffer pointed to by the S parameter.

If the number of bytes specified by the LenS parameter or the number of characters specified by the LenPC parameter are successfully converted, the mbstopcs subroutine returns the number of characters converted, sets the Err parameter to 0, and sets the EndPtr parameter to point immediately after the last character converted in the input buffer pointed to by the S parameter.

If the byte specified by the StopCh parameter is found in the input buffer pointed to by the S parameter, the following occurs:
• Conversion ceases.
• The value specified by the StopCh parameter is placed in the PC parameter.
• The EndPtr parameter is set to point immediately after the value specified by the StopCh parameter.
• The Err parameter is set to 0.
• The number of characters converted is returned.

If an invalid character is found in the input buffer pointed to by the S parameter, the EndPtr parameter is set to point to the start of this character, the Err parameter is set to (size_t)-1, and the mbstopcs subroutine returns the number of characters converted.

If an incomplete character is found at the end of the input buffer pointed to by the S parameter, the EndPtr parameter is set to point to the start of the incomplete character, and the Err parameter is set to the number of bytes in a character starting with the byte pointed to by EndPtr parameter. The mbstopcs subroutine returns the number of characters converted.

The parameters have the following values:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC</td>
<td>Points to a wchar_t array to contain the converted characters.</td>
</tr>
<tr>
<td>LenPC</td>
<td>Specifies the maximum number of wide characters that can be placed in the PC parameter.</td>
</tr>
<tr>
<td>S</td>
<td>Points to a buffer of character data to be converted.</td>
</tr>
<tr>
<td>LenS</td>
<td>Specifies the number of bytes of character data in the S parameter.</td>
</tr>
<tr>
<td>StopCh</td>
<td>Specifies a single-byte character value to indicate end of data in the S parameter.</td>
</tr>
<tr>
<td>EndPtr</td>
<td>Points into the S parameter where character conversion ended.</td>
</tr>
<tr>
<td>Err</td>
<td>Specifies an error value indicating why the conversion failed.</td>
</tr>
</tbody>
</table>

__pctomb Subroutine

The pctomb subroutine converts a process code to a character.

The syntax for the pctomb subroutine is as follows:

```c
size_t pctomb(S, LenS, PC, Err)
char * S;
size_t LenS;
wchar_t * PC;
int * Err;
```

The input buffer pointed to by the PC parameter contains a wide character that the subroutine attempts to convert to a character in the input buffer pointed to by the S parameter. If a valid process code is found in the input buffer pointed to by the PC parameter, it is converted and stored in the input buffer pointed to by the S parameter, and the number of bytes in the character is returned.
If the wide character in the input buffer pointed to by the \textit{PC} parameter is invalid, the \texttt{__pctomb} subroutine returns 0 and sets the \textit{Err} parameter to the value ((\texttt{size_t})-1. If the length of the character is greater than the number of bytes specified by the \textit{LenS} parameter, the \texttt{__pctomb} subroutine returns 0 and sets the \textit{Err} parameter to the number of bytes required to form the character.

The parameters have the following values:

- \textit{S} Points to a buffer to contain the converted process code.
- \textit{LenS} Specifies the size of the character array pointed to by the \textit{S} parameter.
- \textit{PC} Points to the wide character to be converted.
- \textit{Err} Specifies an error value indicating why the conversion failed.

\textbf{\texttt{__pcstombs} Subroutine}

The \texttt{__pcstombs} subroutine converts a wide character string to a character string.

The syntax for the \texttt{__pcstombs} subroutine is as follows:

\begin{verbatim}
size_t __pcstombs(S, LenS, PC, LenPC, StopCh, EndPtr, Err)
char * S;
size_t LenS;
wchar_t * PC;
size_t LenPC;
wchar_t StopCh;
char ** EndPtr;
int * Err;
\end{verbatim}

The input buffer pointed to by the \textit{PC} parameter contains the number of wide characters specified by the \textit{LenPC} parameter. The \texttt{__pcstombs} subroutine attempts to convert the process codes to characters. The conversion continues until one of the following occurs:

- The number of wide characters specified by the \textit{LenPC} parameter have been converted.
- The number of bytes specified by the \textit{LenS} parameter have been converted.
- The character value specified in the \textit{StopCh} parameter is encountered in the input buffer pointed to by the \textit{PC} parameter.
- An invalid wide character is found in the input buffer pointed to by the \textit{PC} parameter.

If the number of bytes specified by the \textit{LenS} parameter or the number of characters specified by the \textit{LenPC} parameter are successfully converted, the \texttt{__pcstombs} subroutine returns the number of bytes placed in the buffer pointed to by the \textit{S} parameter, sets the \textit{Err} parameter to 0, and sets the \textit{EndPtr} parameter to point immediately after the last character converted in the input buffer pointed to by the \textit{PC} parameter.

If the character specified by the \textit{StopCh} parameter is found in the input buffer pointed to by the \textit{PC} parameter, the following occurs:

- Conversion ceases.
- The character specified by the \textit{StopCh} parameter is placed at the end of the data currently pointed to by the \textit{S} parameter.
- The \textit{EndPtr} parameter is set to point immediately after the character specified by the \textit{StopCh} parameter.
- The \textit{Err} parameter is set to 0.
- The number of bytes placed in the buffer pointed to by the \textit{S} parameter is returned.

If an invalid wide character is found in the input buffer pointed to by the \textit{PC} parameter, the \textit{EndPtr} parameter is set to point to the start of this character, the \textit{Err} parameter is set to ((\texttt{size_t})-1, and the \texttt{__pcstombs} subroutine returns the number of bytes placed in the buffer pointed to by the \textit{S} parameter.
The parameters have the following values:

- **S**
  Points to a buffer to contain the converted data.

- **LenS**
  Specifies the size in bytes of the character array pointed to by the **S** parameter.

- **PC**
  Points to a **wchar_t** array to be converted.

- **LenPC**
  Specifies the number of wide characters in the array pointed to by the **PC** parameter.

- **StopCh**
  Specifies a wide-character value to indicate end of data in the array pointed to by the **PC** parameter.

- **EndPtr**
  Points into the **S** parameter where character conversion ended.

- **Err**
  Specifies the error value indicating why the conversion failed.

---

**Files**

```
/usr/lib/nls/loc/*
```

Specifies locale definition source files for supported locales.

```
/usr/lib/nls/charmap/*
```

Specifies character set description (**charmap**) source files for supported locales.

---

**Related Information**

The **locale** command, **localedef** command.

- [Character Set Description (**charmap**) Source File Format](#)
- [Locale Definition Source File Format](#)

For specific information about other locale categories and their keywords, see the **LC_COLLATE** category, **LC_CTYPE** category, **LC_MESSAGES** category, **LC_MONETARY** category, **LC_NUMERIC** and **LC_TIME** category for the locale definition source file format.

- [Changing Your Locale](#)
- [Locale Overview for System Management](#)
- [National Language Support Overview for System Management](#)
- [Understanding the Locale Definition Source File in Operating system and device management](#)

---

**magic File Format**

**Purpose**

Defines file types.

**Description**

The **/etc/magic** file is used by commands such as the following to determine the type of a given file:

- **file** command
- **more** command

Entering the following command would result in a printed message describing the file type of the **FileName** parameter:

```
file FileName
```

If **FileName** contains a byte pattern corresponding to an executable file, the pattern would match a stanza in the **/etc/magic** file and the executable message would be displayed. If the **FileName** is a data file, a data message is displayed, and so on.

The fields of the magic file are as follows:

1. Byte offset
2. Value type
3. Optional relational operator ("=" by default) and value to match (numeric or string constant)
4. String to be printed

Numeric values may be decimal, octal, or hexadecimal. Strings can be entered as hexadecimal values by preceding them with '0x'.

The last string can have one printf format specification.

The > (greater than) symbol in occasional column 1s is magic; it forces commands to continue scanning and matching additional lines. The first line not marked with the > sign terminates the search.

Examples

0 short 2345 this is a dummy type file
0 long 0x1234 this is a different dummy type file
>12 long >0 another possible type
0 short 7895 last type of file

Related Information

The file command, more or page command.

.mailrc File Format

Purpose

Sets defaults for the mail command.

Description

The .mailrc file can be placed in your $HOME directory to personalize the Mail program. You can create the .mailrc file with any ASCII editor. Once the file is created, the Mail program reads the file when you send or read mail, and applies the options you have set. In the file, you can define aliases for other users’ mail addresses. You can also change the way mail is displayed and stored on your system.

The Mail program uses a master file in the same format, /usr/share/lib/Mail.rc. Options you set in your $HOME/.mailrc file override comparable options in the Mail.rc file.

A line that begins with a # (pound sign) followed by a space is treated as a comment. The Mail program ignores the entire line and any entries or options it contains.

Entries

Use the following mail subcommands as entries in the .mailrc file:

mail Subcommand

alias NewAlias { Address... | PreviousAlias... }

ignore FieldList

set [OptionList | Option=Value...]

Definition

Defines an alias or distribution list. The alias can be defined as an actual mail address, or as another alias defined in a previous entry in the .mailrc file. To define a group, enter multiple addresses or previous aliases separated by spaces.

Adds the header fields in the FieldList parameter to the list of fields to be ignored. Ignored fields are not displayed when you look at a message with the Type or print subcommand. Use this subcommand to suppress machine-generated header fields. Use the Type or Print subcommand to print a message in its entirety, including ignored fields.
mail Subcommand

Definition
Sets an option. The argument following the set option can be either an OptionList giving the name of a binary option (an option that is either set or unset) or an Option=Value entry used to assign a value to an option.

unset OptionList

Disables the values of the options specified in OptionList. This action is the inverse of the set OptionList entry.

Binary Options for the set and unset Entries

Use the set entry to enable options and the unset entry to disable options. Add the options you want to set or unset to the $HOME/.mailrc file. The options and the actions they generate are as follows:

append
Adds messages saved in your mailbox to the end rather than to the beginning of the $HOME/mbox file.

ask
Prompts for the subject of each message sent. If you do not wish to create a subject field, press the Enter key at the prompt.

askcc
Prompts for the addresses of people who should receive copies of the message. If you do not wish to send copies, press the Enter key at the prompt.

autoprint
Sets the delete subcommand to delete the current message and display the next message.

dot
Interprets a period entered on a line by itself as the end of a message you are sending.

hold
Holds messages that you have read but have not deleted or saved in the system mailbox instead of in your personal mailbox. This option has no effect on deleted messages.

ignore
Ignores interrupt messages from your terminal and echoes them as @ (at sign) characters.

ignoreeof
Sets the mail command to refuse the Ctrl-D key sequence as the end of a message.

keepsave
Prevents the Mail program from deleting messages that you have saved with the s or w mailbox subcommand. Normally, messages are deleted automatically when you exit the mail command. Use the keepsave and hold options to hold messages in your system mailbox. Otherwise, the messages are placed in your personal mailbox ($HOME/mbox).

metoo
Includes the sender in the alias expansion. By default, expanding the alias removes the sender. When this option is set in your .mailrc file, sending a message using an alias that includes your name sends a copy of the message to your mailbox.

noheader
 Suppresses the list of messages in your mailbox when you start the Mail program. Instead, only the mailbox prompt (&) is displayed. To get a list of messages, use the h mailbox subcommand.

nosave
Prevents retention of interrupted letters in the $HOME/dead.letter file.

quiet
Suppresses the printing of the banner when the Mail program starts. The banner is the line that shows the name of the Mail program.

Replyall
Reverses the meaning of the reply subcommand and the Reply subcommand.

verbose
Displays the actual delivery of messages on the terminal. This is the same as specifying the -v flag on the command line.

Value Options for the set Entry

You can use a set entry to assign values to the following options. For example, enter set screen=20 to limit headers to 20 lines per screen.

crt=Lines
Defines the number of lines of a mail message the Mail program displays before pausing for input (this option starts the pg command to control the scrolling).

EDITOR=Editor
Gives the full path name of the editor to be started with the e mailbox subcommand or the -e mail editor subcommand. The default editor is /usr/bin/e.

escape=Character
Changes the escape character used for mail editor subcommands. The default character is ~ (tilde).

folder=PathName
Gives the path name of a directory in which to store mail folders. Once the directory is defined, you can use the + (plus sign) notation to refer to it when using the FileName parameter with mailbox subcommands.
record=FileName

Defines a file in which to record outgoing mail. The path name must be absolute (that is, a full path name), or be given relative to the current directory.

**Note:** If you set up a file to record outgoing messages, read the file periodically with the `mail -f` command and delete unnecessary messages. Otherwise, the file will grow and eventually use all of your storage space.

screen=Lines

Defines the number of lines of message headers displayed (for example, in response to the `h` mailbox subcommand) before pausing for input.

toplines=Lines

Defines the number of lines displayed by the `top` mailbox subcommand.

VISUAL=Editor

Gives the full path name of the editor to be started with the `v` mailbox subcommand or the `~v` mail editor subcommand. The default editor is `/usr/bin/vi`.

**Examples**

1. To ignore the Message-ID field and the Received field, place the following entry in the `.mailrc` file:
   
   ```
   ignore message-id received
   ```

   When messages are displayed in the mailbox, the machine message ID number and the date your system received the message are not displayed.

2. To set a folder directory, place the following entry in the `.mailrc` file:
   
   ```
   set folder=/home/kaye/notes
   ```

   To save message 1 from the mailbox in the folder procedures, enter the following at the mailbox prompt (`&`):
   
   ```
   s 1 +procedure
   ```

   Message 1 is saved in the `/home/kaye/notes/procedures` file (if the file already exists, the message is appended to the file).

3. To record outgoing mail in a folder directory, place the following pair of entries in the `.mailrc` file:
   
   ```
   set record=/home/pierre/letters/mailout
   set folder=/home/pierre/letters
   ```

   Outgoing mail is placed in the `/home/pierre/letters/mailout` file, and can be read with the following command:
   
   ```
   mail -f +mailout
   ```

4. To combine the delete and print commands and also instruct the Mail program to include your user ID when expanding aliases, enter the following in your `.mailrc` file:
   
   ```
   set autoprint metoo
   ```

   The `autoprint` option causes the next message to be displayed whenever you delete a message. The `metoo` option causes the Mail program to send a copy of messages to you when it expands mail aliases. By default, the Mail program discards your user address when it expands an alias, so that you do not get a copy of mail you send.

5. To unset an option that is set in the `/usr/share/lib/Mail.rc` file, enter the following in your `.mailrc` file:
   
   ```
   unset askcc
   ```

   This entry prevents the mail editor from requesting a carbon copy list when you create messages, even if the `askcc` option is set in the `Mail.rc` file.

6. To set aliases for two users and a distribution list that includes several users, enter the following in your `.mailrc` file:
   
   ```
   alias george george@thor.valhalla.dbm.com
   alias bill Odin.UUCP:@depta.UCCP:@deptb:bill@deptc
   alias mygroup amy@cleo george bill
   ```

   To send mail to user bill using his alias, enter:
To send mail to everyone in the mygroup list, enter:

```
mail mygroup
```

When you complete and send the message, the `mail` command actually addresses it as follows:

```
amy@cleo george@thor.valhalla.dbm.comm odin.UUCP:depta.UCCP:
deptb:bill@deptc
```

**Files**

/`usr/share/lib/Mail.rc` Contains systemwide defaults for the Mail program.

`$HOME/.mailrc` Contains user-specific defaults for the Mail program.

**Related Information**

The `mail` command, `pg` command.

Mail Editor Subcommands for the `mail`, Mail Command

Mailbox Subcommands for the `mail`, Mail Command

Creating and sending mail, Mail program customization options, Mail applications, Mail handling and receiving in Networks and communication management.

---

**map3270 File Format for TCP/IP**

**Purpose**

Defines keyboard mapping and colors for the `tn3270` command.

**Description**

The `/etc/map3270` file defines keyboard mapping and colors for the `tn3270` command. When emulating 3270 terminals, mapping must be performed between key sequences entered on a user’s (ASCII) keyboard and the keys that are available on a 3270 emulator.

For example, the 3270 emulator key `EEOF` erases the contents of the current field from the location of the cursor to the end of the field. In order to accomplish this function, the terminal user and a program emulating a 3270 emulator must be compatible with regard to what keys invoke the `EEOF` function.

The requirements for these sequences are:

- The first character of the sequence is outside of the standard ASCII printable characters.
- No one sequence is an initial part of another (although sequences may share initial parts).

The `/etc/map3270` file consists of entries for various terminals. The first part of an entry lists names of terminals using that entry. These names should be the same as those in the `/usr/share/lib/terminfo/*.ti` files.

**Note:** Often, several terminals from different `/usr/share/lib/terminfo/*.ti` entries use the same `/etc/map3270` file entry. For example, both 925 and 925vb (for 925 with visual bells) might use the same `map3270` file entry. Each name is separated by a `|` (vertical bar), after which comes a `{` (left brace); the definitions; and finally, a `}` (right brace).
Format
The definitions begin with a reserved keyword, which identifies the 3270 function. The keyword is followed by an = (equal sign), which in turn is followed by the various string sequences to generate the particular function. The definitions end with a ; (semi-colon). The string sequences are printable ASCII characters enclosed inside ’’ (single quotes) and separated by | (vertical bars).

Special characters can be used within ’’ (single quotes). A ^ (caret) indicates a control character. For example, the string ’’^a’’ represents Ctrl-A; that is, hexadecimal 1 (the string ’’^A’’ generates the same code). To generate delete or rubout, enter ’’^d’’ ’’^?’ (Ctrl-D or Ctrl-?). To represent a control character in the /etc/map3270 file, you must use the caret. Typing Control-A or Ctrl-A does not work.

Note: The Ctrl-^ key sequence (to generate a hexadecimal 1E) is represented as ’’^^’’ (not ’’\^\’’).

The \ (backslash) special character precedes other characters to change their meaning. Because this has little effect for most characters, its use is not recommended. The backslash prevents a single quote from terminating a string, for example the string ’’^\’’ represents Ctrl-. For a backslash to be part of a string, place two backslashes (’’\’’) in the string.

In addition, the following characters are special:

’’\e’’ Specifies an escape character.
’’\n’’ Specifies a new line.
’’\t’’ Specifies a tab.
’’\r’’ Specifies a carriage return.

It is not necessary for each character in a string to be enclosed within single quotes. The string ’’\e\e\e’’ means three escape characters.

Comments, which may appear anywhere on a line, begin with a # (pound sign) and terminate at the end of that line. However, comments cannot begin inside a quoted string. A pound sign inside a quoted string has no special meaning.

3270 Keys Supported

Note: Some of the following keys do not exist on a 3270 emulator. The functions listed with an * (asterisk) are not supported by the tn3270 command. An unsupported function causes the tn3270 command to send a bell sequence to the user’s terminal.

The /etc/map3270 file supports the following list of 3270 key names:

<table>
<thead>
<tr>
<th>Key Name</th>
<th>Functional Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>altk*</td>
<td>Alternate keyboard dvorak</td>
</tr>
<tr>
<td>aplend*</td>
<td>Treat input as ASCII</td>
</tr>
<tr>
<td>aploff*</td>
<td>APL off</td>
</tr>
<tr>
<td>aplon*</td>
<td>APL on</td>
</tr>
<tr>
<td>attention</td>
<td>Attention key. The attention key sends an IAC BREAK TELNET protocol sequence to the TELNET server on a VM or MVS system. The TELNET server is responsible for implementing the attention key.</td>
</tr>
<tr>
<td>btab</td>
<td>Field tab back</td>
</tr>
<tr>
<td>clear</td>
<td>Local clear of the 3270 screen</td>
</tr>
<tr>
<td>clrtab</td>
<td>Clear all column tabs</td>
</tr>
<tr>
<td>colbak</td>
<td>Column back tab</td>
</tr>
<tr>
<td>coltab</td>
<td>Column tab</td>
</tr>
<tr>
<td>cursel*</td>
<td>Cursor select</td>
</tr>
<tr>
<td>delete</td>
<td>Delete character</td>
</tr>
<tr>
<td>deltab</td>
<td>Delete a column tab</td>
</tr>
<tr>
<td>disc</td>
<td>Disconnect (suspend)</td>
</tr>
<tr>
<td>Key Name</td>
<td>Functional Description</td>
</tr>
<tr>
<td>----------</td>
<td>------------------------</td>
</tr>
<tr>
<td>down</td>
<td>Down cursor</td>
</tr>
<tr>
<td>dp</td>
<td>Duplicate character</td>
</tr>
<tr>
<td>eeof</td>
<td>Erase end of field</td>
</tr>
<tr>
<td>einp</td>
<td>Erase input</td>
</tr>
<tr>
<td>enter</td>
<td>Enter key</td>
</tr>
<tr>
<td>erase</td>
<td>Erase last character</td>
</tr>
<tr>
<td>escape</td>
<td>Enter TELNET command mode</td>
</tr>
<tr>
<td>ferase</td>
<td>Erase field</td>
</tr>
<tr>
<td>fieldend</td>
<td>Tab to last non-blank of current or next unprotected (writable) field</td>
</tr>
<tr>
<td>flinp</td>
<td>Flush input</td>
</tr>
<tr>
<td>fm</td>
<td>Field mark character</td>
</tr>
<tr>
<td>home</td>
<td>Home the cursor</td>
</tr>
<tr>
<td>indent</td>
<td>Indent one tab stop</td>
</tr>
<tr>
<td>init*</td>
<td>New terminal type</td>
</tr>
<tr>
<td>insrt</td>
<td>Toggle insert mode</td>
</tr>
<tr>
<td>left</td>
<td>Left cursor</td>
</tr>
<tr>
<td>lprt*</td>
<td>Local print</td>
</tr>
<tr>
<td>master_reset</td>
<td>Reset, unlock, and redisplay</td>
</tr>
<tr>
<td>nl</td>
<td>New line</td>
</tr>
<tr>
<td>pa1</td>
<td>Program attention 1</td>
</tr>
<tr>
<td>pa2</td>
<td>Program attention 2</td>
</tr>
<tr>
<td>pa3</td>
<td>Program attention 3</td>
</tr>
<tr>
<td>pfk1</td>
<td>Program function key 1</td>
</tr>
<tr>
<td>pfk2</td>
<td>Program function key 2</td>
</tr>
<tr>
<td>.</td>
<td></td>
</tr>
<tr>
<td>.</td>
<td></td>
</tr>
<tr>
<td>.</td>
<td></td>
</tr>
<tr>
<td>pfk36</td>
<td>Program function key 36.</td>
</tr>
<tr>
<td>pcoff*</td>
<td>Xon/xoff off</td>
</tr>
<tr>
<td>pcon*</td>
<td>Xon/xoff on</td>
</tr>
<tr>
<td>reset</td>
<td>Reset key-unlock keyboard</td>
</tr>
<tr>
<td>reshow</td>
<td>Redisplay the screen</td>
</tr>
<tr>
<td>right</td>
<td>Right cursor</td>
</tr>
<tr>
<td>sethom</td>
<td>Set home position</td>
</tr>
<tr>
<td>setmrg</td>
<td>Set left margin</td>
</tr>
<tr>
<td>settab</td>
<td>Set a column tab</td>
</tr>
<tr>
<td>synch</td>
<td>In synch with the user</td>
</tr>
<tr>
<td>tab</td>
<td>Field tab</td>
</tr>
<tr>
<td>treq</td>
<td>Test request</td>
</tr>
<tr>
<td>undent</td>
<td>Undent one tab stop</td>
</tr>
<tr>
<td>up</td>
<td>Up cursor</td>
</tr>
<tr>
<td>werase</td>
<td>Erase last word</td>
</tr>
<tr>
<td>wordbacktab</td>
<td>Tab to beginning of current or last word</td>
</tr>
<tr>
<td>wordend</td>
<td>Tab to end of current or next word</td>
</tr>
<tr>
<td>wordtab</td>
<td>Tab to beginning of next word</td>
</tr>
<tr>
<td>xoff*</td>
<td>Hold output</td>
</tr>
<tr>
<td>xon*</td>
<td>Release output</td>
</tr>
</tbody>
</table>

**A Sample Entry**

The following default entry is included within the `tn3270` command and is used when it is unable to locate a version in the user’s environment or the `/etc/map3270` file.
name {                     # actual name comes from TERM variable
clear = '^z';
flinp = '^x';
enter = '^m';
delete = 'd' | '^?';        # note that '^?' is delete (rubout)
synch = '^r';
reshow = '^v';
eeof = '^e';
tab = '^t';
btab = '^b';

nl = '^n';
left = '^h';
right = '^l';
up = '^k';
down = '^j';
einp = '^w';
reset = '^t';
xoff = '^s';
xon = '^q';
escape = '^c';
ferase = '^u';
isrt = ' ';            # program attention keys
pa1 = '^p1'; pa2 = '^p2'; pa3 = '^p3';
# program function keys
pfk1 = '1'; pfk2 = '2'; pfk3 = '3'; pfk4 = '4';
pfk5 = '5'; pfk6 = '6'; pfk7 = '7'; pfk8 = '8';
pfk9 = '9'; pfk10 = '0'; pfk11 = '-'; pfk12 = '=';
pfk13 = ';'; pfk14 = '0'; pfk15 = '0';
pfk17 = ';'; pfk18 = '0'; pfk19 = ';'; pfk20 = '0';
pfk21 = ' ' pfk22 = '0'; pfk23 = '_'; pfk24 = '0';
}

3270 Key Definitions

The following table shows the proper keys to emulate each 3270 function when using the default key mapping supplied with the tn3270 command.

<table>
<thead>
<tr>
<th>Function</th>
<th>3270 Key</th>
<th>Default Key(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command Keys</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Enter</td>
<td></td>
<td>RETURN</td>
</tr>
<tr>
<td>Clear</td>
<td></td>
<td>Ctrl-z</td>
</tr>
<tr>
<td>Attention</td>
<td></td>
<td>Ctrl-F12</td>
</tr>
<tr>
<td>Cursor Movement Keys</td>
<td></td>
<td></td>
</tr>
<tr>
<td>New line</td>
<td></td>
<td>Ctrl-n or Home</td>
</tr>
<tr>
<td>Tab</td>
<td></td>
<td>Ctrl-i</td>
</tr>
<tr>
<td>Back tab</td>
<td></td>
<td>Ctrl-b</td>
</tr>
<tr>
<td>Cursor left</td>
<td></td>
<td>Ctrl-h</td>
</tr>
<tr>
<td>Cursor right</td>
<td></td>
<td>Ctrl-l</td>
</tr>
<tr>
<td>Cursor up</td>
<td></td>
<td>Ctrl-k</td>
</tr>
<tr>
<td>Cursor down</td>
<td></td>
<td>Ctrl-j or LINE FEED</td>
</tr>
<tr>
<td>Edit Control Keys</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delete char</td>
<td></td>
<td>Ctrl-d or RUB</td>
</tr>
<tr>
<td>Erase EOF</td>
<td></td>
<td>Ctrl-e</td>
</tr>
<tr>
<td>Erase input</td>
<td></td>
<td>Ctrl-w</td>
</tr>
<tr>
<td>Insert mode</td>
<td></td>
<td>ESC Space</td>
</tr>
<tr>
<td>End insert</td>
<td></td>
<td>ESC Space</td>
</tr>
</tbody>
</table>
Program Function Keys

<table>
<thead>
<tr>
<th>Function Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>PF1</td>
</tr>
<tr>
<td>PF2</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>PF10</td>
</tr>
<tr>
<td>PF11</td>
</tr>
<tr>
<td>PF12</td>
</tr>
<tr>
<td>PF13</td>
</tr>
<tr>
<td>PF14</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>PF24</td>
</tr>
<tr>
<td>PF1</td>
</tr>
<tr>
<td>PF2</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>PF10</td>
</tr>
<tr>
<td>PF11</td>
</tr>
<tr>
<td>PF12</td>
</tr>
<tr>
<td>PF13</td>
</tr>
<tr>
<td>PF14</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>PF24</td>
</tr>
</tbody>
</table>

Program Attention Keys

<table>
<thead>
<tr>
<th>Attention Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>PA1</td>
</tr>
<tr>
<td>PA2</td>
</tr>
<tr>
<td>PA3</td>
</tr>
</tbody>
</table>

Local Control Keys

<table>
<thead>
<tr>
<th>Control Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset after error</td>
</tr>
<tr>
<td>Purge input buffer</td>
</tr>
<tr>
<td>Keyboard unlock</td>
</tr>
<tr>
<td>Redisplay screen</td>
</tr>
</tbody>
</table>

Other Keys

<table>
<thead>
<tr>
<th>Other Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Erase current field</td>
</tr>
</tbody>
</table>

Files

<table>
<thead>
<tr>
<th>File Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>/etc/3270.keys</td>
</tr>
<tr>
<td>/usr/share/lib/terminfo/*.ti</td>
</tr>
</tbody>
</table>

Contains the default keyboard mapping.
Files containing terminal information.

Related Information

The **telnet**, **tn**, or **tn3270** command.

The **.3270keys** file format.

Changing the assignment of a key set in *Networks and communication management*.

Maxuusscheds File Format for BNU

**Purpose**

Limits the number of instances of the **uusched** and **uucico** daemons that can run simultaneously.

**Description**

The **/etc/uucp/Maxuusscheds** file limits the number of instances of the Basic Networking Utilities (BNU) **uusched** daemons that can run simultaneously. Since each instance of the **uusched** daemon is associated with one instance of the **uucico** daemon, the file limits the instances of the **uucico** daemon in a similar way. This file is used in conjunction with the lock files in the **/etc/locks** directory to determine the number of systems currently being polled. Use this file to help manage system resources and load averages.
The `Maxuuscheds` file contains an ASCII number that can be changed for your installation. The default is 2. The larger the number, the greater the potential load on the local system. In any case, the limit should always be less than the number of outgoing lines used by BNU.

The `Maxuuscheds` file requires neither configuration nor maintenance, unless the system on which it is installed is contacted frequently and heavily by users on remote systems.

**Files**

- `/etc/locks` directory
  - Contains lock files that prevent multiple uses of devices and multiple calls to systems.

- `/etc/uucp` directory
  - Contains some of the configuration files for BNU, including the `Maxuuscheds` file.

**Related Information**

The `uucico` daemon.

**Configuring BNU** \[Understanding the BNU File and Directory Structure, Understanding the BNU Daemons\] in *Networks and communication management*.

---

**Maxuuxqts File Format for BNU**

**Purpose**

Limits the number of instances of the BNU `uuxqt` daemon that can run simultaneously on the local system.

**Description**

The `/etc/uucp/Maxuuxqts` file limits both the number of instances of the Basic Networking Utilities (BNU) `uuxqt` daemon that can run simultaneously on the local system and the number of commands from remote systems that can run at one time.

This file contains an ASCII number that can be changed for your installation. The default value is 2. The larger the number, the greater the potential load on the local system.

The `Maxuuxqts` file requires neither configuration nor maintenance, unless the system on which it is installed is used frequently and heavily by users on remote systems.

**Files**

- `/etc/uucp` directory
  - Contains some of the configuration files for BNU, including the `Maxuuxqts` file.

**Related Information**

The `uuxqt` daemon.

**Configuring BNU** and `Understanding the BNU File and Directory Structure` in *Networks and communication management*. 
.mh_alias File Format

Purpose
Defines aliases.

Description
An alias file contains lines that associate an alias name with an address or group of addresses. The Message Handler (MH) package reads both personal alias files (customarily the $HOME/.mh_alias file) and a systemwide alias file, the /etc/mh/MailAliases file. Depending on the MH configuration, aliases may also be defined in the /etc/aliases file (see the sendmail command).

The alias file name is an argument to several MH commands. These commands can be set automatically by entries in the .mh_profile file. Personal alias files can have any name, but must follow the format described here. The /etc/mh/MailAliases file is the default alias file for systemwide aliases. This file is set up by a user with root user authority.

Specify your personal alias file in your .mh_profile file. Otherwise, you must use the -alias flag each time you use an MH command that requires this flag.

Each line of an .mh_alias file has one of the following formats:
- Alias: Address-Group
- Alias ; Address-Group
- <Alias-File

The variables are described as follows:

Alias
specifies a simple address.

Address
specifies a simple Internet-style address.

Group
specifies a group name (or number) from the /etc/group file.

Alias-File
specifies a system file name. The MH package treats alias file names as case-sensitive. Alias expansion is case-sensitive as well.

The Address-Group variable can be either of the following:

AddressList
list of addresses that make up a group.

<Alias-File
system file to be read for more alias definitions.

The addresses in the AddressList variable must be separated by commas.

Note: if there are references to aliases within an alias definition, those aliases must be defined in a following line of the alias file.

Special Characters

\ (backslash)
you can continue an alias definition on the next line by ending the line to be continued with a \ (backslash) followed by a new-line character.

< (less than)
if a line starts with a < (less-than sign), MH reads the file specified after the less-than sign for more alias definitions. The reading is done recursively. If an address group starts with a < (less-than sign), MH reads the file specified after the less-than sign and adds the contents of that file to the address list for the alias.
If an address group starts with an = (equal sign), MH consults the /etc/group file for the group specified after an equal sign. The MH package adds each login name occurring as a member of the group to the address list for the alias.

If an address group starts with a + (plus sign), MH consults the /etc/group file to determine the ID of the group. Each login name appearing in the /etc/passwd file that matches the address group is added to the address list for the alias.

If an address group is defined by an * (asterisk), MH consults the /etc/passwd file and adds all login names with a user number greater than 200 (or the value set for everyone in the /etc/mh/mtstailor file) to the address list for the alias.

The following list explains how the system resolves aliases at posting time:

1. The system builds a list of all addresses from the message to be delivered, eliminating duplicate addresses.
2. If the draft originated on the local host, the system performs alias resolution for addresses that have no specified host.
3. For each line in the alias file, the system compares the alias with all existing addresses. If a match is found, the system removes the matched alias from the address list. The system then adds each new address in the address group to the address list. The alias itself is not usually output. Instead, the address group to which the alias maps is output. If the alias is terminated with a ; (semicolon) instead of a : (colon), both the alias and the address are output in the correct form. (This correct form makes replies possible since MH aliases and personal aliases are unknown to the mail transport system.)

In pattern matching, a trailing * (asterisk) in an alias matches just about anything appropriate.

Examples

The following example of an .mh_alias file illustrates some of its features:

```bash
</home/sarah/morealiases
systems:= systems
staff:+ staff
everyone:*
manager: harold@harold
project:lance,mark@remote,peter,manager
```

The first line says that more aliases should be read from the /home/sarah/morealiases file. The systems alias is defined as all users listed as members of the group systems in the /etc/group file. The staff alias is defined as all users whose group ID in the /etc/passwd file is equivalent to the staff group. Finally, the everyone alias is defined as all users with a user ID in the /etc/passwd file greater than 200.

The manager alias is defined as an alias for user harold@harold. The project alias is defined as the users lance, mark@remote, peter, and manager.

Files

- /etc/aliases: Contains systemwide aliases for the sendmail command.
- /etc/group: Contains basic group attributes.
- /etc/passwd: Contains user authentication information.
- /etc/mh/MailAliases: Contains the defaults alias file for systemwide aliases, which is set up by a user with root user authority.
- /etc/mh/mtstailor: Tailors the Message Handler (MH) environment to the local environment.
- .mh_profile: Customizes the Message Handler (MH) package.
Related Information

The `aliases` file, `/etc/group` file, `/etc/passwd` file, `$HOME/.mh_profile` file.

The `ali` command, `conflict` command, `post` command, `sendmail` command, `whom` command.

### mib.defs File Format

#### Purpose

Provides descriptions of Management Information Base (MIB) variables for the `snmpinfo` command.

#### Description

The `mib.defs` file provides object descriptions of MIB variables for the `snmpinfo` command issued with the `get`, `next`, `set`, and `dump` options. See the `snmpinfo` command for more information. This command is part of Simple Network Management Protocol Agent Applications in Network Support Facilities.

The `mib.defs` file is not intended to be edited by the user. The file should be created with the `mosy` command. See the `mosy` command for information on how to create the `mib.defs` file. This file has the following format:

The MIB group fields are separated by spaces or tabs and contain the following information:

- **GroupDescriptor**: Holds the textual name of the MIB group.
- **GroupEntry**: Denotes the parent MIB group and the location of this MIB group in the parent group. This field is used by the `snmpinfo` command to resolve the ASN.1 dotted notation for MIB variables under this group.

The MIB groups are defined as follows:

<table>
<thead>
<tr>
<th>Group Descriptor</th>
<th>Group Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>internet</td>
<td>iso.3.6.1</td>
</tr>
<tr>
<td>directory</td>
<td>internet.1</td>
</tr>
<tr>
<td>mgmt</td>
<td>internet.2</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>mib-2</td>
<td>mgmt.1</td>
</tr>
<tr>
<td>system</td>
<td>mib-2.1</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
</tr>
</tbody>
</table>

The MIB variable fields are separated by spaces or tabs, and contain the following information:

- **ObjectDescriptor**: Holds the textual name of the object.
**GroupEntry**
Denotes the MIB object group and the location of this MIB variable in this group. This field is used by the `snmpinfo` command to resolve the ASN.1 dotted notation for this MIB variable.

**Syntax**
Denotes the type of the object as one of the following:
- INTEGER
- OCTET STRING or DisplayString
- OBJECT IDENTIFIER
- Network Address
- Counter
- Gauge
- TimeTicks
- Opaque

**Access**
Designates the access permissions for the object and can be one of the following:
- Read-only
- Read-write
- Write-only
- Not-accessible

**Status**
Designates the RFC 1213 compliance status of the object and can be one of the following:
- Mandatory
- Optional
- Deprecated
- Obsolete

The parent MIB group definition required for a particular MIB variable `GroupEntry` definition must precede the object definition for the MIB variable.

Comments begin with a # (pound sign) or - - (two dashes) and continue to the end of the line.

**Files**

`/usr/samples/snmpd/smi.my`
Defines the ASN.1 definitions by which the SMI is defined as in RFC 1155.

`/usr/samples/snmpd/mibII.my`
Defines the ASN.1 definitions for the MIB II variables as defined in RFC 1213.

**Related Information**
The `mosy` command, `snmpinfo` command.

Understanding the Management Information Base (MIB) in *AIX 5L Version 5.3 Communications Programming Concepts*.

**named.conf File Format for TCP/IP**

**Purpose**
Defines the configuration and behavior of the `named` daemon.
Description

The /etc/named.conf file is the default configuration file for the named and named9 server. If the named daemon is started without specifying an alternate file, the named daemon reads this file for information on how to set up the local name server.

The format of the named.conf file will be different depending on which version of the named server is configured. File format information for both named8 and named9 can be found below.

Note: The named daemon reads the configuration file only when the named daemon starts or when the named daemon receives an SRC refresh command or a SIGHUP signal.

The data in the named.conf file specifies general configuration characteristics for the name server, defines each zone for which the name server is responsible (its zones of authority), and provides further config information per zone, possibly including the source DOMAIN database file for the zone.

Any database files referenced in the named.conf file must be in Standard Resource Record Format. These data files can have any name and any directory path. However, for convenience in maintaining the named database, they are generally given names in the following form: /etc/named.extension. The general format of named data files is described in DOMAIN Data File, DOMAIN Reverse Data File, DOMAIN Cache File, and DOMAIN Local File.

Format of the named.conf file when configuring named8

General

Comments in the named.conf file can begin with a # (pound sign) or // (two forward slashes), or can be enclosed in the C-style comment characters, e.g., /* comment text */.

Configuration options are lines of text beginning with a keyword, possibly including some option text or a list, and ending in a ; (semicolon).

The named.conf file is organized into stanzas. Each stanza is an enclosed set of configuration options that define either general characteristics of the daemon or a zone configuration. Certain stanza definitions are allowed only at the top-level, therefore nesting these stanzas is not allowed. The current top-level configuration stanza keywords are: acl, key, logging, options, server, and zone.

Further configuration information can be incorporated into the conf file via the include keyword. This keyword directs the daemon to insert the contents of the indicated file into the current position of the include directive.

Access Control List (ACL) Definition

acl acl-name {
    access-element;
    [ access-element; ... ]
};

Defines an access control list to be referenced throughout the configuration file by acl-name. Multiple acl definitions can exist within one configuration file provided that each acl-name is unique. Additionally, four default access control lists are defined:
- any Any host is allowed.
- none No host is allowed.
- localhost Only the localhost is allowed.
- localnets Only hosts on a network matching a name server interface is allowed.
<table>
<thead>
<tr>
<th>Option</th>
<th>Values</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>access-element</td>
<td>IP-address</td>
<td>Defines a source as allowed or disallowed. Multiple access-elements are allowed inside the acl stanza.</td>
</tr>
<tr>
<td></td>
<td>IP-prefix</td>
<td></td>
</tr>
<tr>
<td></td>
<td>acl-reference</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Each element can be an IP address in dot notation (e.g., 9.3.149.66) an IP prefix in CIDR or slash notation (e.g., 9.3.149/24) or a reference to another access control list (e.g., localhost).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Additionally, each element indicates whether the element is allowed or disallowed access via an ! (exclamation point) modifier prepended to the element.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For example: acl hostlist1 { !9.53.150.239; 9.3.149/24; 9.3.149/24; }</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When the access control list &quot;hostlist1&quot; is referenced in the configuration, it implies to allow access from any host whose IP address begins with 9.3.149 and to disallow access from the internet host 9.53.150.239.</td>
</tr>
</tbody>
</table>

**Key Definition**

```plaintext
key key-name {
    algorithm alg-id;
    secret secret-string;
};
```

Defines an algorithm and shared secret key to be referenced in a server stanza and used for authentication by that name server. This feature is included for future use and is currently unused in the name server.

<table>
<thead>
<tr>
<th>Option</th>
<th>Values</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>algorithm</td>
<td>alg-id string</td>
<td>A quoted-string that defines the type of security algorithm that will be used when interpreting the secret string. None are defined at this time.</td>
</tr>
<tr>
<td>secret</td>
<td>secret-string</td>
<td>A quoted-string that is used by the algorithm to authenticate the host.</td>
</tr>
</tbody>
</table>

**Logging Configuration**

```plaintext
logging {
    [ channel channel-name {
        file file-name
        [ versions ( num-vers | unlimited ) ]
        [ size size-value ]
        syslog ( kern | user | mail | daemon | syslog | lpr | news | uucp )
    }
};
```

Chapter 2. File Formats 481
In this newest version of the name server, the logging facility has been greatly improved to allow for much reconfiguration of the default logging mechanism. The **logging** stanza is used to define logging output channels and to associate the predefined logging categories with either the predefined or user-defined logging output channels.

When no logging stanza is included in the conf file, the name server still logs messages and errors just as it has in previous releases. Informational and some critical messages will be logged through the syslog daemon facility, and debug and other esoteric information will be logged to the **named.run** file when the global debug level (set with the `-d` command-line option) is non-zero.

<table>
<thead>
<tr>
<th>Option</th>
<th>Values</th>
<th>Explanation</th>
</tr>
</thead>
</table>
| channel  |                             | Defines an output channel to be referenced later by the `channel-name` identifier. An output channel specifies a destination for output messages to be sent as well as some formatting information to be used when writing the output message. More than one output channel can be defined provided that each `channel-identifier` is unique. Also, each output channel can be referenced from multiple logging categories. There are four predefined output channels:  
  - `default_syslog` sends “info” and higher severity messages to syslog’s “daemon” facility  
  - `default_debug` writes debug messages to the **named.run** file as specified by the global debug level  
  - `default_stderr` writes “info” and higher severity messages to stderr  
  - **null** discards all messages |

<p>| file     | <code>file-name</code> string          | Defines an output channel as one that logs messages to an output file. The file used for output is specified with the <code>file-name</code> string. Additionally, the <strong>file</strong> option allows for controlling how many versions of the output file should be kept, and what size limit the output file should never exceed. The <strong>file</strong>, <strong>syslog</strong>, and <strong>null</strong> output paths are mutually exclusive. |</p>
<table>
<thead>
<tr>
<th>Option</th>
<th>Values</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>versions</td>
<td>num-versions unlimited</td>
<td>Specifies the number of old output files that should be kept. When an output file is reopened, rather than replacing a possible existing output file, the existing output file will be saved as an old output file with a .value extension. Using the num-versions value, one can limit the number of old output files to be kept. However, specifying the unlimited keyword indicates to continually accumulate old output file versions. By default, no old versions of any log file are kept.</td>
</tr>
<tr>
<td>size</td>
<td>size-value</td>
<td>Specifies the maximum size of the log file used by this channel. By default, the size is unlimited. However, when a size is configured, once size-value bytes are written to the file, nothing more will be written until the file is reopened. Accepted values for size-value include the word “unlimited” and numbers with k, m, or g modifiers specifying kilobytes, megabytes, and gigabytes respectively. For example, 1000k and 1m indicate one thousand kilobytes and one megabyte respectively.</td>
</tr>
<tr>
<td>syslog</td>
<td>kern user mail daemon auth syslog lpr news uucp</td>
<td>Defines an output channel as one that redirects its messages to the syslog service. The supported value keywords correspond to facilities logged by the syslog service. Ultimately, the syslog service will define which received messages will be logged through the service, therefore, if defining a channel to redirect its messages to the syslog service’s user facility would not result in any visibly logged messages if the syslog service is not configured to output messages from this facility. For more information concerning the syslog service, see the <a href="https://www.kernel.org/pub/linux/libs/log/syslog/">syslogd</a> daemon. The file, syslog, and null output paths are mutually exclusive.</td>
</tr>
<tr>
<td>null</td>
<td></td>
<td>Defines an output channel through which all messages will be discarded. All other output channel options are invalid for an output channel whose output path is null.</td>
</tr>
<tr>
<td>Option</td>
<td>Values</td>
<td>Explanation</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>severity</td>
<td>critical</td>
<td>Sets a threshold of message severities to be logged through the output channel. While these severity definitions are similar to those used by the syslog service, for the name server they also control output through file path channels. Messages must meet or exceed the severity level to be logged through the output channel. The dynamic severity specifies that the name server's global debug level (specified when the daemon is invoked with the -d flag) controls which messages pass through the output channel. Also, the debug severity can specify a level modifier which is an upper threshold for debug messages whenever the name server has debugging enabled at any level. A lower debug level indicates less information is to be logged through the channel. It is not necessary for the global debug level to meet or exceed the debug level value.</td>
</tr>
<tr>
<td></td>
<td>error</td>
<td></td>
</tr>
<tr>
<td></td>
<td>warning</td>
<td></td>
</tr>
<tr>
<td></td>
<td>notice</td>
<td></td>
</tr>
<tr>
<td></td>
<td>info</td>
<td></td>
</tr>
<tr>
<td></td>
<td>debug [ level ]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>dynamic</td>
<td></td>
</tr>
<tr>
<td>Option</td>
<td>Values</td>
<td>Explanation</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>print-category</td>
<td>yes</td>
<td>Controls the format of the output message when it is sent through the output path. Regardless of which, how many, or in which order these options are listed inside the channel stanza, the message will be prepended with the the text in a time, category, severity order.</td>
</tr>
<tr>
<td>print-severity</td>
<td>no</td>
<td>The following is an example of a message with all three print- options enabled:</td>
</tr>
<tr>
<td></td>
<td>no</td>
<td>By default, no extra text will be prepended to an output message.</td>
</tr>
<tr>
<td>category</td>
<td>yes</td>
<td>Note that when the syslog service logs messages, it also prepends the date and time information to the text of the message. Thus, enabling print-time on a channel that uses the syslog output path would result in the syslog service logging a message with two dates prepended to it.</td>
</tr>
<tr>
<td></td>
<td>no</td>
<td></td>
</tr>
</tbody>
</table>

The category keyword defines a stanza which associates a logging or messaging category with predefined or user-defined output channels.

By default, the following categories are defined:

```plaintext
category default {
    default_syslog;
    default_debug;
};

category panic {
    default_syslog;
    default_debug;
};
```
<table>
<thead>
<tr>
<th>Option</th>
<th>Values</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>category-name</td>
<td>default</td>
<td>The category-name specifies which logging category is to be associated with the listed channel-references. This results in any output text generated by the name server daemon for that logging category to be redirected through each of the channel-references listed.</td>
</tr>
<tr>
<td></td>
<td>config</td>
<td></td>
</tr>
<tr>
<td></td>
<td>parser</td>
<td></td>
</tr>
<tr>
<td></td>
<td>queries</td>
<td></td>
</tr>
<tr>
<td></td>
<td>lame-servers</td>
<td></td>
</tr>
<tr>
<td></td>
<td>statistics</td>
<td></td>
</tr>
<tr>
<td></td>
<td>panic</td>
<td></td>
</tr>
<tr>
<td></td>
<td>update</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ncache</td>
<td></td>
</tr>
<tr>
<td></td>
<td>xfer-in</td>
<td></td>
</tr>
<tr>
<td></td>
<td>xfer-out</td>
<td></td>
</tr>
<tr>
<td></td>
<td>db</td>
<td></td>
</tr>
<tr>
<td></td>
<td>event-lib</td>
<td></td>
</tr>
<tr>
<td></td>
<td>packet</td>
<td></td>
</tr>
<tr>
<td></td>
<td>notify</td>
<td></td>
</tr>
<tr>
<td></td>
<td>cname</td>
<td></td>
</tr>
<tr>
<td></td>
<td>security</td>
<td></td>
</tr>
<tr>
<td></td>
<td>os</td>
<td></td>
</tr>
<tr>
<td></td>
<td>insist</td>
<td></td>
</tr>
<tr>
<td></td>
<td>maintenance</td>
<td></td>
</tr>
<tr>
<td></td>
<td>load</td>
<td></td>
</tr>
<tr>
<td></td>
<td>response-checks</td>
<td></td>
</tr>
<tr>
<td>channel-reference</td>
<td></td>
<td>References a channel-name identifier defined previously in the logging configuration stanza. Therefore, every message associated with the defined category-name will be logged through each of the defined channel-references.</td>
</tr>
</tbody>
</table>

**Global Options**

```plaintext
options {
    [ directory path-string; ]
    [ named-xfer path-string; ]
    [ dump-file path-string; ]
    [ pid-file path-string; ]
    [ statistics-file path-string; ]
    [ auth-nxdomain ( yes | no ); ]
    [ fake-iquery ( yes | no ); ]
    [ fetch-glue ( yes | no ); ]
    [ multiple-cnames ( yes | no ); ]
    [ notify ( yes | no ); ]
    [ recursion ( yes | no ); ]
    [ forward ( only | first ); ]
    [ forwarders ( ipaddr; [...] ); ]
    [ check-names
        ( master|slave|response )
        ( warn|fail|ignore ); ]
    [ allow-query { access-element; [...] }; ]
    [ allow-transfer { access-element; [...] }; ]
    [ listen-on [ port port-num ] { access-element; [...] }; ... ]
    [ query-source [ address ( ipaddr|* ) ] [ port ( port|* ) ]; ]
    [ max-transfer-time-in seconds; ]
    [ transfer-format ( one-answer | many-answers ); ]
    [ transfers-in value; ]
    [ transfers-out value; ]
```
Defines many globally available options to modify basic characteristics of the name server.

Because some of the options in this configuration stanza may modify the behavior in how the named daemon will read and interpret later sections of the named file, it is highly recommended that the options stanza be the first stanza listed in the configuration file.

<table>
<thead>
<tr>
<th>Option</th>
<th>Values</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>directory</td>
<td>path-string</td>
<td>&quot;.&quot;</td>
</tr>
<tr>
<td>named-xfer</td>
<td>path-string</td>
<td>&quot;/usr/sbin/named-xfer&quot;</td>
</tr>
<tr>
<td>dump-file</td>
<td>path-string</td>
<td>&quot;/usr/tmp/named_dump.db&quot;</td>
</tr>
<tr>
<td>pid-file</td>
<td>path-string</td>
<td>&quot;/etc/named.pid&quot;</td>
</tr>
<tr>
<td>statistics-file</td>
<td>path-string</td>
<td>&quot;/usr/tmp/named.stats&quot;</td>
</tr>
<tr>
<td>auth-nxdomain</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>fake-iquery</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Option</td>
<td>Values</td>
<td>Default</td>
</tr>
<tr>
<td>---------------------</td>
<td>--------</td>
<td>---------</td>
</tr>
<tr>
<td>fetch-glue</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>multiple-cnames</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>notify</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>recursion</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>forward</td>
<td>only</td>
<td></td>
</tr>
<tr>
<td></td>
<td>first</td>
<td>first</td>
</tr>
<tr>
<td>forwarders</td>
<td>ipaddr</td>
<td>{empty list}</td>
</tr>
<tr>
<td>Option</td>
<td>Values</td>
<td>Default</td>
</tr>
<tr>
<td>---------------------</td>
<td>----------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td><strong>check-names</strong></td>
<td>master ignore</td>
<td>master fail</td>
</tr>
<tr>
<td></td>
<td>master warn</td>
<td>slave warn</td>
</tr>
<tr>
<td></td>
<td>master fail</td>
<td>response ignore</td>
</tr>
<tr>
<td></td>
<td>slave ignore</td>
<td>slave warn</td>
</tr>
<tr>
<td></td>
<td>slave warn</td>
<td>response warn</td>
</tr>
<tr>
<td></td>
<td>response ignore</td>
<td>response fail</td>
</tr>
<tr>
<td><strong>allow-query</strong></td>
<td>access-element</td>
<td>any</td>
</tr>
<tr>
<td><strong>allow-transfer</strong></td>
<td>access-element</td>
<td>any</td>
</tr>
<tr>
<td><strong>listen-on</strong></td>
<td>port port-num</td>
<td>port 53 { localhost; }</td>
</tr>
<tr>
<td></td>
<td>address access-element</td>
<td></td>
</tr>
<tr>
<td><strong>query-source</strong></td>
<td>address ipaddr</td>
<td>address * port *</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
<td>Type</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>max-transfer-time-in</td>
<td>Specifies the maximum amount of time an inbound zone transfer will be allowed to run before it is aborted. This is used to control an event in which a child process of the name server does not execute or terminate properly.</td>
<td>seconds</td>
</tr>
<tr>
<td>transfer-format</td>
<td>Controls the method in which full zone transfers will be sent to requestors. The one-answer method uses one packet per zone resource record while many-answers will insert as many resource records into one packet as possible. While the many-answers method is more efficient, it is only understood by the newest revisions of the name server. This option can be overridden in the server stanza to specify the method on a per name server basis.</td>
<td>one-answer</td>
</tr>
<tr>
<td>transfers-in</td>
<td>Specifies the maximum number of concurrent inbound zone transfers. While this will limit the amount of time each slave zone is out of synchronization with the master’s database, because each inbound transfer runs in a separate child process, increasing the value may also increase the load on the slave server.</td>
<td>value</td>
</tr>
<tr>
<td>transfers-out</td>
<td>Specifies the maximum number of concurrent outbound zone transfers for the name server. This option is currently unused in the server, but will be available at a later time.</td>
<td>value</td>
</tr>
<tr>
<td>transfers-per-ns</td>
<td>Specifies the maximum amount of concurrent zone transfers from a specific remote name server. While this will limit the amount of time each slave zone is out of synchronization with the master’s database, increasing this value may increase the load on the remote master server.</td>
<td>value</td>
</tr>
<tr>
<td>coresize</td>
<td>Configures some process specific values for the daemon. The default values or those inherited by the system and by the system’s resources. Each size-value can be specified as a number or as a number followed by the k, m, and g modifiers indicating kilobytes, megabytes, and gigabytes respectively.</td>
<td>size-value</td>
</tr>
<tr>
<td>datasize</td>
<td>See coresize.</td>
<td>size-value</td>
</tr>
<tr>
<td>files</td>
<td>See coresize.</td>
<td>value</td>
</tr>
<tr>
<td>stacksize</td>
<td>size-value</td>
<td>default</td>
</tr>
<tr>
<td>--------------------</td>
<td>--------------</td>
<td>---------</td>
</tr>
<tr>
<td>See coresize.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**clean-interval**

Controls the intervals for the periodic maintenance tasks of the name server.

The clean-interval specifies how frequently the server will remove expired resource records from the cache. The interface-interval specifies how frequently the server will rescan for interfaces in the system. The statistics-interval specifies how frequently the name server will output statistics data.

A minutes value of zero indicates that the service task should only run when the configuration file is reread.

<table>
<thead>
<tr>
<th>clean-interval</th>
<th>minutes</th>
<th>60</th>
</tr>
</thead>
</table>

**interface-interval**

See clean-interval.

<table>
<thead>
<tr>
<th>interface-interval</th>
<th>minutes</th>
<th>60</th>
</tr>
</thead>
</table>

**statistics-interval**

See clean-interval.

<table>
<thead>
<tr>
<th>statistics-interval</th>
<th>minutes</th>
<th>60</th>
</tr>
</thead>
</table>

**cleandb-time**

Specifies a time of day in which the database will be scanned and any dynamic records whose set of SIG resource records are all expired will be removed. For a dynamic zone which has update-security set to presecured, only the expired SIG KEY will remain.

The default is to never perform this scan. Instead, the expired records will remain until the name is queried.

**time** is specified as HH:MM in a 24-hour format.

<table>
<thead>
<tr>
<th>cleandb-time</th>
<th>time</th>
<th>N/A</th>
</tr>
</thead>
</table>

**topology**

Specifies a search order to use to find a preference in a list of addresses corresponding to a name server. Whenever a query is forwarded or a query must be made to another name server, it may be necessary to choose an address from a list of available addresses.

Each access-element, while seemingly similar to those specified in an acl stanza, is interpreted by its position in the list. The first elements in the list are preferred more than those following them. Negated elements (those specified with the ! (exclamation point) modifier) are considered least desirable.

<table>
<thead>
<tr>
<th>topology</th>
<th>access-element</th>
<th>localhost; localnets;</th>
</tr>
</thead>
</table>

**Server Specific Options**

**server ipaddr**

```plaintext
{ [ bogus (yes | no)];
 [ transfers value;
```
transfer-format ( one-answer | many-answers ); }

Modifies the behavior in which the remote name server matching the specified \textit{ipaddr} IP address should be treated.

<table>
<thead>
<tr>
<th>Option</th>
<th>Values</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>bogus</td>
<td>yes, no</td>
<td>Indicates that the name server identified by the stanza should not be used again. The default value is \textit{no}.</td>
</tr>
<tr>
<td>transfers</td>
<td>\textit{value}</td>
<td>Overrides the globally available option \textit{transfers-per-ns}. Specifies a maximum value for the number of concurrent inbound zone transfers from the foreign name server identified by the stanza.</td>
</tr>
<tr>
<td>transfer-format</td>
<td>one-answer, many-answers</td>
<td>Overrides the globally available option \textit{transfer-format} to a specific value for the specified server. The \textit{transfer-format} option indicates to the name server how to form its outbound full zone transfers. By default, the value is inherited from the \textit{options} stanza (where it defaults to \textit{one-answer}). \textit{one-answer} specifies that only one resource record can be sent per packet during the zone transfer, whereas \textit{many-answers} indicates to entirely fill the outbound packet with resource records. The \textit{many-answers} format is only available in the newest revisions of the name server.</td>
</tr>
</tbody>
</table>

Zone Definition

```
zone \textit{domain-string} \textbf{[ class ]} {
  type ( hint | stub | slave | master );
  [ file \textit{path-string}; ]
  [ masters \{ \textit{ipaddr}; [... ] \}; ]
  [ check-names ( warn | fail | ignore ); ]
  [ allow-update \{ access-element; [... ] \}; ]
  [ allow-query \{ access-element; [... ] \}; ]
  [ allow-transfer \{ access-element; [... ] \}; ]
  [ max-transfer-time-in seconds; ]
  [ notify \{ yes | no \}; ]
  [ also-notify \{ \textit{ipaddr}; [... ] \}; ]
  [ dont-notify \{ \textit{ipaddr}; [... ] \}; ]
  [ notify-delaytime seconds; ]
  [ notify-retrytime seconds; ]
  [ notify-retrycount value; ]
  [ dump-interval seconds; ]
  [ incr-interval seconds; ]
  [ deferupdcnt value; ]
  [ key-xfer \{ yes | no \}; ]
  [ timesync \{ yes | no \}; ]
  [ timesync-xfer \{ yes | no \}; ]
```

Zone Definition
The zone stanza is used to define a zone, its type, possible location of data, and operating parameters. The `domain-string` is a quoted string specifying the zone, where "." is used to specify the root zone. The `class` parameter specifies the `class` of the zone as either `in`, `hs`, `hesiod`, or `chaos`. By default, the `class` is assumed to be `IN`.

<table>
<thead>
<tr>
<th>Option and Description</th>
<th>Values</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>Defines the type of the zone. Hint zones, previously regarded as cache zones, only describe a source for information not contained in the other defined zones. A stub zone is one similar to a slave zone. While the slave zone replicates the entire database of its master, the stub zone only replicates the NS resource records. The master zone maintains a database on disk. Based upon the selection of zone type, some of the other options are required while others may be impertinent. Zones of type hint and master require the file option, while zones of type slave and stub require the masters option. Additionally, the only other option available to a hint zone is the check-names option.</td>
<td>hint, stub, slave, master</td>
</tr>
<tr>
<td>file</td>
<td>Specifies the location for the source of data specific to the zone. This parameter is only optional for stub and slave zones, where its inclusion indicates that a locally saved copy of the remote zone can be kept. The <code>path-string</code> parameter is a quoted string which can specify the file name either non-relative or relative to the <code>options</code> stanza's directory. If the path is intended to be specified relative to the server root, the <code>options</code> stanza must be specified before the zone stanza.</td>
<td><code>path-string</code></td>
</tr>
<tr>
<td>masters</td>
<td>Specifies a list of sources that will be referenced for a slave or stub zone to retrieve its data. This option is not valid for any other type of zone, and must be included for either of these two types.</td>
<td><code>ipaddr</code></td>
</tr>
<tr>
<td>check-names</td>
<td>Overrides the check-names option in the global <code>options</code> stanza. The default value is inherited from the <code>options</code> stanza, where its default is <code>fail</code> for master zones and <code>warn</code> for slave zones.</td>
<td>warn, fail, ignore</td>
</tr>
<tr>
<td>Option and Description</td>
<td>Values</td>
<td>Default</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>allow-update</td>
<td>access-element</td>
<td>none</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>update-security</td>
<td>unsecured, presecured,</td>
<td>unsecured</td>
</tr>
<tr>
<td></td>
<td>controlled</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>allow-query</td>
<td>access-element</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>allow-transfer</td>
<td>access-element</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>max-transfer-time-in</td>
<td>seconds</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>notify</td>
<td>yes, no</td>
<td></td>
</tr>
</tbody>
</table>

**allow-update**
Indicates from what source addresses a zone will accept dynamic updates. *access-elements* are specified in the same manner as they are for the `acl` stanza. Because of the inherent insecurity of a dynamic update, this value defaults to `none`. If no `update-security` is specified, dynamic updates should be limited to a specific set of secured machines.

**update-security**
Valid only when the `allow-update` option specifies at least one source address, `update-security` defines what type of secured update mechanism the zone will use. The current zone update security method is a non-standard two-key method, but is compatible with previous releases of the name server.

- `presecured` indicates that a zone will only accept updates for which names and resource records already exist, unless the update is signed by the zone’s authorizing key. Normally, this means that the zone must be prepopulated with the names and records it is to maintain. `controlled` specifies a zone in which names can be added to the database without the signature of the zone’s authorizing key, but existing records cannot be modified without being signed by the `KEY` resource record’s corresponding private key.

Note that a proper `presecured` or `controlled` zone must contain a zone `KEY` resource record. See the [TCP/IP name resolution](#) for more information regarding zone update security.

**allow-query**
Overrides the globally available option `allow-query`. This option's default is inherited from the global `options` stanza, where its default is any.

**allow-transfer**
Overrides the globally available option `allow-transfer`. This option's default is inherited from the global `options` stanza, where its default is any.

**max-transfer-time-in**
Overrides the globally available option `max-transfer-time-in`. This option's default is inherited from the global `options` stanza, where its default is 120.

**notify**
Overrides the globally available option `notify`. This option's default is inherited from the global `options` stanza, where its default is `yes`.
<table>
<thead>
<tr>
<th>Option and Description</th>
<th>Values</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>also-notify</td>
<td>ipaddr</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>The default NOTIFY mechanism will notify slave servers of a change in the DOMAIN database in order to limit the amount of time that the slave server retains a zone out of synchronization with the master server. The also-notify option allows for the addition of addresses to submit the notifications.</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dont-notify</td>
<td>ipaddr</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Specifies a list of IP addresses to be removed from the default list of NOTIFY recipients. This option is useful if a name server is known to be problematic when receiving NOTIFY requests.</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>notify-delaytime</td>
<td>seconds</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Specifies an estimated time of delay between notifications to multiple name servers. Because the receipt of a NOTIFY message usually triggers the prompt request for a zone transfer, this option can tune to latency in which each server will respond with the request for the modified zone. The real value used will be randomized between the specified number of seconds and twice this value.</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>notify-retrytime</td>
<td>seconds</td>
<td>60</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Specifies the number of seconds in which the name server will wait to retransmit a NOTIFY message which has gone unresponded.</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>notify-retrycount</td>
<td>value</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Specifies the maximum number of tries that the name server will attempt to send unanswered NOTIFY messages to other name servers.</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dump-interval</td>
<td>seconds</td>
<td>3600</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| **Specifies an interval in which the name server will rewrite a dynamic zone to the zone file. In the interim, all updates and other transactions will be logged in the transaction log file for performance reasons. Aside from this periodic zone dump, the transaction log file is only discarded and the zone is only dumped when the name server is properly shut down. This option is only valid for zones in which the allow-update option specifies at least one valid accessor. **Note:** The transaction log file name is the zone file name with an appended “.log” extension.**
<table>
<thead>
<tr>
<th>Option and Description</th>
<th>Values</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>incr-interval</td>
<td>seconds</td>
<td>300</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Specifies an interval in which the name server will accept dynamic updates while not increasing the zone’s SOA record’s serial level. Because a change in the zone SOA record will instantiate a NOTIFY message, limiting this occurrence will limit the amount of zone transfer requests at the expense of minimal zone differences between a dynamic master server and its slave. This option is only valid for zones in which the allow-update option specifies at least one valid accessor.</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>deferupdcnt</td>
<td>value</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Specifies a threshold value for the number of properly applied updates received during one incr-interval interval. If more than value updates are realized during the interval, the name server will modify the zone SOA serial level and subsequently NOTIFY each of the slave servers. Use this value to limit the database replication inconsistencies in an environment where dynamic zone updates occur infrequently but in large magnitude. This option is only valid for zones in which the allow-update option specifies at least one valid accessor.</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>key-xfer</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Specifies whether the server should transmit KEY resource records during a zone transfer. In a very controlled environment where KEY queries will only be made to the master name server, setting this option to no will save zone transfer time and improve performance.</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>timesync</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Specifies that a name server should calculate the true expiration time of a SIG resource record using its own clock rather than relying on the expiration time set by a possible update source. This removes the inconsistencies involved when dynamic zone updaters have their system clocks misaligned from the name server host. Because enabling this option modifies the output and interpretation of a SIG resource record in a DOMAIN database file, disabling this option may be required when manually transferring a DOMAIN database file to another name server.</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>timesync-xfer</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Specifies which SIG resource record expiration time will be transferred during a zone transfer. Enabling this option is only valid when the timesync option is enabled.</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Option and Description

<table>
<thead>
<tr>
<th>Option</th>
<th>Values</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ixfr-directory</code></td>
<td><code>path-string</code></td>
<td></td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specifies a directory in which temporary data files will be contained for use with this zone. The datafiles contain incremental zone changes and are essential to the proper use of the Incremental Zone Transfer (IXFR) method. Because these files are created and destroyed dynamically by the name server, one should not specify a globally-writable directory. Additionally, the directory specified must be unique from other <code>ixfr-directory</code> options specified in other zones. The default value for this directory is derived from the zone's file name or domain name. By default, a directory is created in an “ixfrdata” directory within the name server’s default directory. Contained in this directory will be subdirectory matching the base name of the zone’s file name or domain name. It is not necessary to specify this option for the proper behavior of the IXFR feature.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| `save-backups`          | `yes`  | `no`    |
| **Description**         |        |
| To properly calculate an incremental zone difference between server invocations, it is necessary to determine the zone database differences prior to the shutdown of the server and after the loading of the server. By enabling this option, a backup of the zone file will be written and read upon loading of the name server to determine any zone differences. While enabling this option is necessary to use the IXFR transfer method after a stop and restart transition of the name server, it is not necessary to realize incremental zone differences when a zone file is modified and signalled to reload via the SRC refresh command or SIGHUP signal. |

| `separate-dynamic`      | `yes`  | `no`    |
| **Description**         |        |
| Instructs the name server to retain $INCLUDE references in a dynamic zone when the DOMAIN database file is written to disk. The behavior of this feature implies that resource records that can be modified through the dynamic update mechanism exist in the DOMAIN database file referenced by the file option, while other resource records that should not be modified through the dynamic update mechanism be contained in files included (through the $INCLUDE directive) by the DOMAIN database file. |

### Examples

The following examples show the some of the various ways to use configure a simple named.conf file. In these examples, two networks are represented: abc and xyz.

Network abc consists of:
• gobi.abc, the master name server for the abc network, 192.9.201.2
• mojave.abc, a host machine, 192.9.201.6
• sandy.abc, a slave name server for the abc network and the gateway between abc and xyz, 192.9.201.3

Network xyz consists of:
• kalahari.xyz, master name server for the xyz network, 160.9.201.4
• lopnor.xyz, a host machine, 160.9.201.5
• sahara.xyz, a host machine and hint name server for the xyz network, 160.9.201.13
• sandy.xyz, a slave name server for the xyz network and gateway between abc and xyz, 160.9.201.3

Note: sandy, a gateway host, is on both networks and also serves as a slave name server for both domains.

1. The /etc/named.conf file for gobi.abc, the master name server for network abc, contains these entries:

```
# conf file for abc master server - gobi.abc
#
server 192.9.201.3 {
    transfer-format many-answers;
};
zone "abc" in {
    type master;
    file "/etc/named.abcdata";
    allow-update { localhost; }
};
zone "201.9.192.in-addr.arpa" in {
    type master;
    file "/etc/named.abcrev";
    allow-update { localhost; }
};
zone "0.0.127.in-addr.arpa" in {
    type master;
    file "/etc/named.abclocal";
};
```

2. The /etc/named.conf file for kalahari.xyz, the master name server for network xyz, contains these entries:

```
# conf file for abc master server - kalahari.xyz
#
# acl xyz-slaves {
#    160.9.201.3;
#};
# options {
#    directory "/etc";
#    allow-transfer { xyz-slaves; localhost; }
#};
zone "xyz" in {
    type master;
    file "named.xyzdata";
};
```
zone "9.160.in-addr.arpa" in {
  type master;
  file "named.xyxrev";
};
zone "0.0.127.in-addr.arpa" in {
  type master;
  file "named.xyzlocal";
};

3. The `/etc/named.conf` file for sandy, the slave name server for networks abc and xyz, contains the following entries:

```bash
# conf file for slave server for abc and xyz - sandy
# options {
  directory "/etc";
};
zone "abc" in {
  type slave;
  masters { 192.9.201.2; };
  file "named.abcdata.bak";
};
zone "xyz" in {
  type slave;
  masters { 160.9.201.4; };
  file "named.xyzdata.bak";
};
zone "201.9.192.in-addr.arpa" in {
  type slave;
  masters { 192.9.201.2; };
};
zone “9.160.in-addr.arpa” in {
  type slave;
  masters { 192.9.201.4; };
};
zone “0.0.127.in-addr.arpa” in {
  type master;
  file "named.local";
};
```

4. The `/etc/named.conf` file for sahara, a hint name server for the network xyz, contains the following entries:

```bash
# conf file for hint server for xyz - sahara
# zone "." in {
  type hint;
  file "/etc/named.ca";
};
zone "0.0.127.in-addr.arpa" in {
  type master;
  file "/etc/named.local";
};
```
Format of the named.conf file when configuring named9

General
A BIND 9 configuration consists of statements and comments. Statements end with a semicolon. Statements and comments are the only elements that can appear without enclosing braces. Many statements contain a block of substatements, which are also terminated with a semicolon.

The following statements are supported:

<table>
<thead>
<tr>
<th>Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>acl</td>
<td>defines a named IP address matching list, for access control and other uses.</td>
</tr>
<tr>
<td>controls</td>
<td>declares control channels to be used by the rndc utility.</td>
</tr>
<tr>
<td>include</td>
<td>includes a file.</td>
</tr>
<tr>
<td>key</td>
<td>specifies key information for use in authentication and authorization using TSIG.</td>
</tr>
<tr>
<td>logging</td>
<td>specifies what the server logs, and where the log messages are sent.</td>
</tr>
<tr>
<td>options</td>
<td>controls global server configuration options and sets defaults for other statements.</td>
</tr>
<tr>
<td>server</td>
<td>sets certain configuration options on a per-server basis.</td>
</tr>
<tr>
<td>trusted-keys</td>
<td>defines trusted DNSSEC keys.</td>
</tr>
<tr>
<td>view</td>
<td>defines a view.</td>
</tr>
<tr>
<td>zone</td>
<td>defines a zone.</td>
</tr>
</tbody>
</table>

The logging and options statements may only occur once per configuration.

acl Statement Grammar

```
_acl acl-name {
  address_match_list
};
```

acl Statement Definition and Usage

The acl statement assigns a symbolic name to an address match list. It gets its name from a primary use of address match lists: Access Control Lists (ACLs).

Note that an address match list's name must be defined with acl before it can be used elsewhere; no forward references are allowed.

The following ACLs are built-in:

<table>
<thead>
<tr>
<th>ACL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>any</td>
<td>Matches all hosts.</td>
</tr>
<tr>
<td>none</td>
<td>Matches no hosts.</td>
</tr>
<tr>
<td>localhost</td>
<td>Matches the IPv4 addresses of all network interfaces on the system.</td>
</tr>
<tr>
<td>localnets</td>
<td>Matches any host on an IPv4 network for which the system has an interface.</td>
</tr>
</tbody>
</table>

The localhost and localnets ACLs do not currently support IPv6 (that is, localhost does not match the host’s IPv6 addresses, and localnets does not match the host’s attached IPv6 networks) due to the lack of a standard method of determining the complete set of local IPv6 addresses for a host.

controls Statement Grammar

```
controls {
  inet { ip_addr | * } [ port ip_port ] allow { address_match_list }
  keys { key_list };
  [ inet ...; ]
};
```
controls Statement Definition and Usage

The controls statement declares control channels to be used by system administrators to affect the operation of the local nameserver. These control channels are used by the rndc utility to send commands to and retrieve non-DNS results from a nameserver.

An inet control channel is a TCP/IP socket accessible to the Internet, created at the specified ip_port on the specified ip_addr. If no port is specified, port 953 is used by default. "*" cannot be used for ip_port.

The ability to issue commands over the control channel is restricted by the allow and keys clauses. Connections to the control channel are permitted based on the address permissions in address_match_list. key_id members of the address_match_list are ignored, and instead are interpreted independently based the key_list. Each key_id in the key_list is allowed to be used to authenticate commands and responses given over the control channel by digitally signing each message between the server and a command client. All commands to the control channel must be signed by one of its specified keys to be honored.

If no controls statement is present, named9 will set up a default control channel listening on the loopback address 127.0.0.1 and its IPv6 counterpart ::1. In this case, and also when the controls statement is present but does not have a keys clause, named9 will attempt to load the command channel key from the /etc/rndc.key file in (or whatever sysconfdir was specified as when BIND was built). To create a rndc.key file, run rndc-confgen -a.

The rndc.key feature was created to ease the transition of systems from BIND 8, which did not have digital signatures on its command channel messages and thus did not have a keys clause. It makes it possible to use an existing BIND 8 configuration file in BIND 9 unchanged, and still have rndc work the same way ndc worked in BIND 8, simply by executing the command rndc-keygen -a after BIND 9 is installed.

Since the rndc.key feature is only intended to allow the backward-compatible usage of BIND 8 configuration files, this feature does not have a high degree of configurability. You cannot easily change the key name or the size of the secret, so you should make a rndc.conf with your own key if you wish to change those things. The rndc.key file also has its permissions set such that only the owner of the file (the user that named is running as) can access it. If you desire greater flexibility in allowing other users to access rndc commands then you need to create an rndc.conf and make it group readable by a group that contains the users who should have access. The UNIX control channel type of BIND 8 is not supported in BIND 9. If it is present in the controls statement from a BIND 8 configuration file, it is ignored and a warning is logged.

include Statement Grammar

include filename;

include Statement Definition and Usage

The include statement inserts the specified file at the point that the include statement is encountered. The include statement facilitates the administration of configuration files by permitting the reading or writing of some things but not others. For example, the statement could include private keys that are readable only by a nameserver.

key Statement Grammar

key key_id {
    algorithm string;
    secret string;
};

key Statement Definition and Usage

The key statement defines a shared secret key for use with TSIG.
The `key` statement can occur at the top level of the configuration file or inside a view statement. Keys defined in top-level `key` statements can be used in all views. Keys intended for use in a controls statement must be defined at the top level.

The `key_id`, also known as the key name, is a domain name uniquely identifying the key. It can be used in a "server" statement to cause requests sent to that server to be signed with this key, or in address match lists to verify that incoming requests have been signed with a key matching this name, algorithm, and secret. The `algorithm_id` is a string that specifies a security/authentication algorithm. The only algorithm currently supported with TSIG authentication is `hmac-md5`. The `secret_string` is the secret to be used by the algorithm, and is treated as a base-64 encoded string.

logging Statement Grammar

```
logging {
    [ channel channel_name {
        file path name
        [ versions ( number | unlimited ) ]
        [ size size spec ]
        syslog syslog_facility
        stderr
        null ;
        [ severity ( critical | error | warning | notice |
                    info | debug [ level ] | dynamic ); ]
        [ print-category yes or no ; ]
        [ print-severity yes or no ; ]
        [ print-time yes or no ; ]
    } ; ]
    [ category category_name {
        channel_name ; [ channel_name ; ... ]
    } ; ]
    ...
};
```

logging Statement Definition and Usage

The `logging` statement configures a wide variety of logging options for the nameserver. Its `channel` phrase associates output methods, format options and severity levels with a name that can then be used with the `category` phrase to select how various classes of messages are logged.

Only one `logging` statement is used to define as many channels and categories as are wanted. If there is no `logging` statement, the logging configuration will be:

```
logging {
    category "unmatched" { "null" ; }
    category "default" { "default_syslog" ; "default_debug" ; }
};
```

In BIND 9, the logging configuration is only established when the entire configuration file has been parsed. In BIND 8, it was established as soon as the `logging` statement was parsed. When the server is starting up, all logging messages regarding syntax errors in the configuration file go to the default channels, or to standard error if the `-g` option was specified.

The channel Phrase

All log output goes to one or more channels; you can make as many of them as you want.

Every channel definition must include a destination clause that says whether messages selected for the channel go to a file, to a particular `syslog` facility, to the standard error stream, or are discarded. It can optionally also limit the message severity level that will be accepted by the channel (the default is info), and whether to include a named-generated time stamp, the category name and/or severity level (the default is not to include any).
The **null** destination clause causes all messages sent to the channel to be discarded; in that case, other options for the channel are meaningless.

The **file** destination clause directs the channel to a disk file. It can include limitations both on how large the file is allowed to become, and how many versions of the file will be saved each time the file is opened.

If you use the **versions** log file option, then **named9** will retain that many backup versions of the file by renaming them when opening. For example, if you choose to keep 3 old versions of the file lamers.log then just before it is opened lamers.log.1 is renamed to lamers.log.2, lamers.log.0 is renamed to lamers.log.1, and lamers.log is renamed to lamers.log.0. You can say versions unlimited; to not limit the number of versions. If a size option is associated with the log file, then renaming is only done when the file being opened exceeds the indicated size. No backup versions are kept by default; any existing log file is simply appended.

The **size** option for files is used to limit log growth. If the file ever exceeds the size, then **named9** will stop writing to the file unless it has a **versions** option associated with it. If backup versions are kept, the files are rolled as described above and a new one begun. If there is no **versions** option, no more data will be written to the log until some out-of-band mechanism removes or truncates the log to less than the maximum size. The default behavior is not to limit the size of the file.

Example usage of the size and versions options:
```plaintext
channel "an_example_channel" {
  file "example.log" versions 3 size 20m;
  print-time yes;
  print-category yes;
};
```

The **syslog** destination clause directs the channel to the system log. Its argument is a **syslog** facility as described in the syslog man page. How **syslog** will handle messages sent to this facility is described in the syslog.conf man page. If you have a system which uses a very old version of syslog that only uses two arguments to the openlog() function, then this clause is silently ignored. The severity clause works like syslog's "priorities," except that they can also be used if you are writing straight to a file rather than using syslog. Messages which are not at least of the severity level given will not be selected for the channel; messages of higher severity levels will be accepted.

If you are using syslog, then the syslog.conf priorities will also determine what eventually passes through. For example, defining a channel facility and severity as daemon and debug but only logging daemon.warning via syslog.conf will cause messages of severity info and notice to be dropped. If the situation were reversed, with named9 writing messages of only warning or higher, then syslogd would print all messages it received from the channel.

The **stderr** destination clause directs the channel to the server's standard error stream. This is intended for use when the server is running as a foreground process, for example when debugging a configuration.

The server can supply extensive debugging information when it is in debugging mode. If the server's global debug level is greater than zero, then debugging mode will be active. The global debug level is set either by starting the named9 server with the -d flag followed by a positive integer, or by running rndc trace. The global debug level can be set to zero, and debugging mode turned off, by running ndc notrace. All debugging messages in the server have a debug level, and higher debug levels give more detailed output. Channels that specify a specific debug severity, for example:
```plaintext
channel "specific_debug_level" {
  file "foo";
  severity debug 3;
};
```
will get debugging output of level 3 or less any time the server is in debugging mode, regardless of the
global debugging level. Channels with dynamic severity use the server's global level to determine what
messages to print.

If print-time has been turned on, then the date and time will be logged. print-time may be specified for a
syslog channel, but is usually pointless since syslog also prints the date and time. If print-category is
requested, then the category of the message will be logged as well. Finally, if print-severity is on, then
the severity level of the message will be logged. The print-ptions may be used in any combination, and
will always be printed in the following order: time, category, severity. Here is an example where all three
print- options are on:
28-Feb-2000 15:05:32.863 general: notice: running

There are four predefined channels that are used for named9's default logging as follows.

```
channel "default_syslog" {
    syslog daemon;
    // send to syslog's daemon
    severity info;
    // facility
    // only send priority info
    // and higher
};
```

```
channel "default_debug" {
    file "named.run";
    // write to named.run in
    // the working directory
    // Note: stderr is used instead
    // of "named.run"
    // if the server is started
    // with the '-f' option.
    severity dynamic;
    // log at the server's
    // current debug level
};
```

```
channel "default_stderr" {
    stderr;
    severity info;
    // only send priority info
    // and higher
};
```

```
channel "null" {
    null;
    // toss anything sent to
    // this channel
};
```

The default_debug channel has the special property that it only produces output when the server's debug
level is nonzero. It normally writes to a file named9run in the server's working directory.

For security reasons, when the -u command line option is used, the named9run file is created only after
named9 has changed to the new UID, and any debug output generated while named9 is starting up and
still running as root is discarded. If you need to capture this output, you must run the server with the -g
option and redirect standard error to a file.

Once a channel is defined, it cannot be redefined. Thus you cannot alter the built-in channels directly, but
you can modify the default logging by pointing categories at channels you have defined.

The category Phrase
There are many categories, so you can send the logs you want to see wherever you want, without seeing
logs you don’t want. If you don’t specify a list of channels for a category, then log messages in that
category will be sent to the default category instead. If you don’t specify a default category, the following
"default default" is used:
As an example, let's say you want to log security events to a file, but you also want keep the default logging behavior. You'd specify the following:

```plaintext
channel "my_security_channel" { 
    file "my_security_file"; 
    severity info; 
};
category "security" { 
    "my_security_channel"; 
    "default_syslog"; 
    "default_debug"; 
};
```

To discard all messages in a category, specify the null channel:

```plaintext
category "xfer-out" { "null"; }; 
category "notify" { "null"; }; 
```

Following are the available categories and brief descriptions of the types of log information they contain.

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>default</td>
<td>The default category defines the logging options for those categories where no specific configuration has been defined.</td>
</tr>
<tr>
<td>general</td>
<td>The catch-all. Many things still aren't classified into categories, and they all end up here.</td>
</tr>
<tr>
<td>database</td>
<td>Messages relating to the databases used internally by the name server to store zone and cache data.</td>
</tr>
<tr>
<td>security</td>
<td>Approval and denial of requests.</td>
</tr>
<tr>
<td>config</td>
<td>Configuration file parsing and processing.</td>
</tr>
<tr>
<td>resolver</td>
<td>DNS resolution, such as the recursive lookups performed on behalf of clients by a caching name server.</td>
</tr>
<tr>
<td>xfer-in</td>
<td>Zone transfers the server is receiving.</td>
</tr>
<tr>
<td>xfer-out</td>
<td>Zone transfers the server is sending.</td>
</tr>
<tr>
<td>notify</td>
<td>The NOTIFY protocol.</td>
</tr>
<tr>
<td>client</td>
<td>Processing of client requests.</td>
</tr>
<tr>
<td>unmatched</td>
<td>Messages that named was unable to determine the class of or for which there was no matching view. A one line summary is also logged to the client category. This category is best sent to a file or stderr, by default it is sent to the null channel.</td>
</tr>
<tr>
<td>network</td>
<td>Network operations.</td>
</tr>
<tr>
<td>update</td>
<td>Dynamic updates.</td>
</tr>
<tr>
<td>queries</td>
<td>Queries. Using the category queries will enable query logging.</td>
</tr>
<tr>
<td>dispatch</td>
<td>Dispatching of incoming packets to the server modules where they are to be processed.</td>
</tr>
<tr>
<td>dnssec</td>
<td>DNSSEC and TSIG protocol processing.</td>
</tr>
<tr>
<td>lame-servers</td>
<td>Lame servers. These are misconfigurations in remote servers, discovered by BIND 9 when trying to query those servers during resolution.</td>
</tr>
</tbody>
</table>

**options Statement Grammar**

```plaintext
options { 
    [ version version_string; ] 
    [ directory path_name; ] 
    [ named-xfer path_name; ] 
    [ tkey-domain domainname; ] 
    [ tkey-dhkey key_name key_tag; ] 
}
```
options Statement Definition and Usage

The options statement sets up global options to be used by BIND. This statement may appear only once in a configuration file. If more than one occurrence is found, the first occurrence determines the actual options used, and a warning will be generated. If there is no options statement, an options block with each option set to its default will be used.

version

The version the server should report via a query of name version.bind in class CHAOS. The default is the real version number of this server.

directory

The working directory of the server. Any non-absolute pathnames in the configuration file will be taken as relative to this directory. The default location for most server output files (e.g. named.run) is this directory. If a directory is not specified, the working directory defaults to ",", the directory from which the server was started. The directory specified should be an absolute path.

named-xfer

This option is obsolete. It was used in BIND 8 to specify the pathname to the named-xfer program. In BIND 9, no separate named-xfer program is needed; its functionality is built into the name server.

tkey-domain

The domain appended to the names of all shared keys generated with TKEY. When a client requests a TKEY exchange, it may or may not specify the desired name for the key. If present, the name of the shared key will be client specified part + tkey-domain. Otherwise, the name of the shared key will be random hex digits + tkey-domain. In most cases, the domainname should be the server’s domain name.

tkey-dhkey

The Diffie-Hellman key used by the server to generate shared keys with clients using the Diffie-Hellman mode of TKEY. The server must be able to load the public and private keys from files in the working directory. In most cases, the keyname should be the server’s host name.

dump-file

The pathname of the file the server dumps the database to when instructed to do so with rndc dumpdb. If not specified, the default is named_dump.db.

memstatistics-file

The pathname of the file the server writes memory usage statistics to on exit. If not specified, the default is named.memstats.

Note: Not yet implemented in BIND 9.

pid-file

The pathname of the file the server writes its process ID in. If not specified, the default is /var/run/named.pid. The pid-file is used by programs that want to send signals to the running nameserver.
statistics-file
The pathname of the file the server appends statistics to when instructed to do so using rndc stats. If not specified, the default is named.stats in the server’s current directory.

port
The UDP/TCP port number the server uses for receiving and sending DNS protocol traffic. The default is 53. This option is mainly intended for server testing; a server using a port other than 53 will not be able to communicate with the global DNS.

random-device
The source of entropy to be used by the server. Entropy is primarily needed for DNSSEC operations, such as TKEY transactions and dynamic update of signed zones. This option specifies the device (or file) from which to read entropy. If this is a file, operations requiring entropy will fail when the file has been exhausted. If not specified, the default value is /dev/random (or equivalent) when present, and none otherwise. The random-device option takes effect during the initial configuration load at server startup time and is ignored on subsequent reloads.

Boolean Options

auth-nxdomain
If yes, then the AA bit is always set on NXDOMAIN responses, even if the server is not actually authoritative. The default is no; this is a change from BIND 8. If you are using very old DNS software, you may need to set it to yes.

deallocate-on-exit
This option was used in BIND 8 to enable checking for memory leaks on exit. BIND 9 ignores the option and always performs the checks.

dialup
If yes, the server treats all zones as if they are doing zone transfers across a dial on demand dialup link, which can be brought up by traffic originating from this server. This has different effects according to zone type and concentrates the zone maintenance so that it all happens in a short interval, once every heartbeat-interval and hopefully during the one call. It also suppresses some of the normal zone maintenance traffic. The default is no. The dialup option may also be specified in the view and zone statements, in which case it overrides the global dialup option.

If the zone is a master zone then the server will send out a NOTIFY request to all the slaves. This will trigger the zone serial number check in the slave (providing it supports NOTIFY) allowing the slave to verify the zone while the connection is active.

If the zone is a slave or stub zone, then the server will suppress the regular “zone up to date” (refresh) queries and only perform them when the heartbeat-interval expires in addition to sending NOTIFY requests.

Finer control can be achieved by using notify, which only sends NOTIFY messages; notify-passive, which sends NOTIFY messages and suppresses the normal refresh queries; and refresh, which suppresses normal refresh processing and send refresh queries when the heartbeat-interval expires and passive which just disables normal refresh processing.

fake-iquery
In BIND 8, this option was used to enable simulating the obsolete DNS query type IQUERY. BIND 9 never does IQUERY simulation.

fetch-glue
This option is obsolete. In BIND 8, fetch-glue yes caused the server to attempt to fetch glue resource records it didn’t have when constructing the additional data section of a response. This is now considered bad practice, and BIND 9 never does it.

has-old-clients
This option was incorrectly implemented in BIND 8, and is ignored by BIND 9. To achieve the intended effect of has-old-clients yes, specify the two separate options auth-nxdomain yes and rfc2308-type1 no instead.
**host-statistics**

In BIND 8, this enables keeping of statistics for every host that the nameserver interacts with. It is not implemented in BIND 9.

**maintain-ixfr-base**

This option is obsolete. It was used in BIND 8 to determine whether a transaction log was kept for Incremental Zone Transfer. BIND 9 maintains a transaction log whenever possible. If you need to disable outgoing incremental zone transfers, use `provide-ixfr no`.

**minimal-responses**

If yes, then when generating responses the server will only add records to the authority and additional data sections when they are required. This may improve the performance of the server. The default is no.

**multiple-cnames**

This option was used in BIND 8 to allow a domain name to allow multiple CNAME records in violation of the DNS standards. BIND 9.2 strictly enforces the CNAME rules both in master files and dynamic updates.

**notify**

If yes (default), DNS NOTIFY messages are sent when a zone the server is authoritative for changes. The messages are sent to the servers listed in the zone’s NS records (except the master server identified in the SOA MNAME field), and to any servers listed in the `also-notify` option. If explicit, notifies are sent only to servers explicitly listed using `also-notify`. If no, no notifies are sent.

The `notify` option may also be specified in the `zone` statement, in which case it overrides the `options notify` statement. It would only be necessary to turn off this option if it caused slaves to crash.

**recursion**

If yes, and a DNS query requests recursion, then the server will attempt to do all the work required to answer the query. If recursion is off and the server does not already know the answer, it will return a referral response. The default is yes.

*Note:* Setting `recursion no` does not prevent clients from getting data from the server’s cache; it only prevents new data from being cached as an effect of client queries. Caching may still occur as an effect the server’s internal operation, such as NOTIFY address lookups.

**rfc2308-type1**

Setting this to yes will cause the server to send NS records along with the SOA record for negative answers. The default is no.

*Note:* Not yet implemented in BIND 9.

**use-id-pool**

This option is obsolete. BIND 9 always allocates query IDs from a pool.

**zone-statistics**

If yes, the server will, by default, collect statistical data on all zones in the server. These statistics may be accessed using `rndc stats`, which will dump them to the file listed in the `statistics-file`. See "The Statistics File" on page 517.

**use-ixfr**

This option is obsolete. If you need to disable IXFR to a particular server or servers see the information on the `provide-ixfr` option in "server Statement Definition and Usage" on page 517.

**provide-ixfr**

See the description of `provide-ixfr` in "server Statement Definition and Usage" on page 517.

**request-ixfr**

See the description of `request-ixfr` in "server Statement Definition and Usage" on page 517.
treat-cr-as-space
This option was used in BIND 8 to make the server treat carriage return ("\r") characters the same way as a space or tab character to facilitate loading of zone files on a UNIX system that were generated on a Windows NT or DOS machine. In BIND 9, both UNIX "\n" and NT/DOS "\r\n" newlines are always accepted, and the option is ignored.

additional-from-auth, additional-from-cache
These options control the behavior of an authoritative server when answering queries which have additional data, or when following CNAME and DNAME chains.
When both of these options are set to yes (default) and a query is being answered from authoritative data (a zone configured into the server), the additional data section of the reply will be filled in using data from other authoritative zones and from the cache. In some situations this is undesirable, such as when there is concern over the correctness of the cache, or in servers where slave zones may be added and modified by untrusted third parties. Also, avoiding the search for this additional data will speed up server operations at the possible expense of additional queries to resolve what would otherwise be provided in the additional section.

For example, if a query asks for an MX record for host foo.example.com, and the record found is "MX 10 mail.example.net", normally the address records for mail.example.net will be provided as well, if known. Setting these options to no disables this behavior.

These options are intended for use in authoritative-only servers, or in authoritative-only views. Attempts to set them to no without also specifying recursion no; will cause the server to ignore the options and log a warning message.

Specifying additional-from-cache no actually disables the use of the cache not only for additional data lookups but also when looking up the answer. This is usually the desired behavior in an authoritative-only server where the correctness of the cached data is an issue.

When a name server is non-recursively queried for a name that is not below the apex of any served zone, it normally answers with an “upwards referral” to the root servers or the servers of some other known parent of the query name. Since the data in an upwards referral comes from the cache, the server will not be able to provide upwards referrals when additional-from-cache no has been specified. Instead, it will respond to such queries with REFUSED. This should not cause any problems since upwards referrals are not required for the resolution process.

match-mapped-addresses
If yes, then an IPv4-mapped IPv6 address will match any address match list entries that match the corresponding IPv4 address. Enabling this option is sometimes useful on IPv6-enabled Linux systems, to work around a kernel quirk that causes IPv4 TCP connections such as zone transfers to be accepted on an IPv6 socket using mapped addresses, causing address match lists designed for IPv4 to fail to match. The use of this option for any other purpose is discouraged.

Forwarding
The forwarding facility can be used to create a large site-wide cache on a few servers, reducing traffic over links to external nameservers. It can also be used to allow queries by servers that do not have direct access to the Internet, but wish to look up exterior names anyway. Forwarding occurs only on those queries for which the server is not authoritative and does not have the answer in its cache.

forward
This option is only meaningful if the forwarders list is not empty. A value of first, the default, causes the server to query the forwarders first, and if that doesn’t answer the question the server will then look for the answer itself. If only is specified, the server will only query the forwarders.

forwarders
Specifies the IP addresses to be used for forwarding. The default is the empty list (no forwarding).

Forwarding can also be configured on a per-domain basis, allowing for the global forwarding options to be overridden in a variety of ways. You can set particular domains to use different forwarders, or have a different forward only/first behavior, or not forward at all. See zone Statement Grammar on page 520.
Access Control
Access to the server can be restricted based on the IP address of the requesting system.

allow-notify
Specifies which hosts are allowed to notify slaves of a zone change in addition to the zone masters. The allow-notify option may also be specified in the zone statement, in which case it overrides the options allow-notify statement. It is only meaningful for a slave zone. If not specified, the default is to allow notify messages only from a zone’s master.

allow-query
Specifies which hosts are allowed to ask ordinary questions. The allow-query option may also be specified in the zone statement, in which case it overrides the options allow-query statement. If not specified, the default is to allow queries from all hosts.

allow-recursion
Specifies which hosts are allowed to make recursive queries through this server. If not specified, the default is to allow recursive queries from all hosts. Note that disallowing recursive queries for a host does not prevent the host from retrieving data that is already in the server’s cache.

allow-transfer
Specifies which hosts are allowed to receive zone transfers from the server. allow-transfer may also be specified in the zone statement, in which case it overrides the options allow-transfer statement. If not specified, the default is to allow transfers from all hosts.

blackhole
Specifies a list of addresses that the server will not accept queries from or use to resolve a query. Queries from these addresses will not be responded to. The default is none.

Interfaces
The interfaces and ports that the server will answer queries from may be specified using the listen-on option. listen-on takes an optional port, and an address_match_list. The server will listen on all interfaces allowed by the address match list. If a port is not specified, port 53 will be used.

Multiple listen-on statements are allowed. For example:

listen-on { 5.6.7.8; };
listen-on port 1234 { 11.2.3.4; 1.2/16; };

This will enable the nameserver on port 53 for the IP address 5.6.7.8, and on port 1234 of an address on the machine in net 1.2 that is not 1.2.3.4.

If no listen-on is specified, the server will listen on port 53 on all interfaces.

Query Address
If the server doesn’t know the answer to a question, it will query other nameservers. query-source specifies the address and port used for such queries. If address is * or is omitted, a wildcard IP address (INADDR_ANY) will be used. If port is * or is omitted, a random unprivileged port will be used. The defaults are as follows:

query-source address * port *;
query-source-v6 address * port *

Note: The address specified in the query-source option is used for both UDP and TCP queries, but the port applies only to UDP queries. TCP queries always use a random unprivileged port.

Zone Transfers
BIND has mechanisms in place to facilitate zone transfers and set limits on the amount of load that transfers place on the system. The following options apply to zone transfers.

also-notify
Defines a global list of IP addresses of name servers that are also sent NOTIFY messages
whenever a fresh copy of the zone is loaded, in addition to the servers listed in the zone’s NS records. This helps to ensure that copies of the zones will quickly converge on stealth servers. If an also-notify list is given in a zone statement, it will override the options also-notify statement. When a zone notify statement is set to no, the IP addresses in the global also-notify list will not be sent NOTIFY messages for that zone. The default is the empty list (no global notification list).

max-transfer-time-in
Inbound zone transfers running longer than this many minutes will be terminated. The default is 120 minutes (2 hours).

max-transfer-idle-in
Inbound zone transfers making no progress in this many minutes will be terminated. The default is 60 minutes (1 hour).

max-transfer-time-out
Outbound zone transfers running longer than this many minutes will be terminated. The default is 120 minutes (2 hours).

max-transfer-idle-out
Outbound zone transfers making no progress in this many minutes will be terminated. The default is 60 minutes (1 hour).

serial-query-rate
Slave servers will periodically query master servers to find out if zone serial numbers have changed. Each such query uses a minute amount of the slave server’s network bandwidth. To limit the amount of bandwidth used, BIND 9 limits the rate at which queries are sent. The value of the serial-query-rate option, an integer, is the maximum number of queries sent per second. The default is 20.

serial-queries
In BIND 8, the serial-queries option set the maximum number of concurrent serial number queries allowed to be outstanding at any given time. BIND 9 does not limit the number of outstanding serial queries and ignores the serial-queries option. Instead, it limits the rate at which the queries are sent as defined using the serial-query-rate option.

transfer-format
Zone transfers can be sent using two different formats, one-answer and many-answers. The transfer-format option is used on the master server to determine which format it sends. one-answer uses one DNS message per resource record transferred. many-answers packs as many resource records as possible into a message. many-answers is more efficient, but is only supported by relatively new slave servers, such as BIND 9, BIND 8.x and patched versions of BIND 4.9.5. The default is many-answers. transfer-format may be overridden on a per-server basis by using the server statement.

transfers-in
The maximum number of inbound zone transfers that can be running concurrently. The default value is 10. Increasing transfers-in may speed up the convergence of slave zones, but it also may increase the load on the local system.

transfers-out
The maximum number of outbound zone transfers that can be running concurrently. Zone transfer requests in excess of the limit will be refused. The default value is 10.

transfers-per-ns
The maximum number of inbound zone transfers that can be concurrently transferring from a given remote nameserver. The default value is 2. Increasing transfers-per-ns may speed up the convergence of slave zones, but it also may increase the load on the remote nameserver. transfers-per-ns may be overridden on a per-server basis by using the transfers phrase of the server statement.
transfer-source
  _transfer-source_ determines which local address will be bound to IPv4 TCP connections used to
  fetch zones transferred inbound by the server. It also determines the source IPv4 address, and
  optionally the UDP port, used for the refresh queries and forwarded dynamic updates. If not set, it
  defaults to a system controlled value which will usually be the address of the interface "closest to"
  the remote end. This address must appear in the remote end’s _allow-transfer_ option for the zone
  being transferred, if one is specified. This statement sets the _transfer-source_ for all zones, but
  can be overridden on a per-view or per-zone basis by including a _transfer-source_ statement
  within the _view_ or _zone_ block in the configuration file.

notify-source
  _notify-source_ determines which local source address, and optionally UDP port, will be used to
  send NOTIFY messages. This address must appear in the slave server’s _masters zone_ clause or
  in an _allow-notify_ clause. This statement sets the _notify-source_ for all zones, but can be
  overridden on a per-zone / per-view basis by including a _notify-source_ statement within the _zone
  or _view_ block in the configuration file.

Operating System Resource Limits
The server’s usage of many system resources can be limited. Scaled values are allowed when specifying
resource limits. For example, 1G can be used instead of _1073741824_ to specify a limit of one gigabyte.
The _unlimited_ option requests unlimited use, or the maximum available amount. The _default_ option uses
the limit that was in force when the server was started.

The following options set operating system resource limits for the name server process. Some operating
systems don’t support some or any of the limits. On such systems, a warning will be issued if the
unsupported limit is used.

coresize
  The maximum size of a core dump. The default is _default_.

data-size
  The maximum amount of data memory the server may use. The default is _default_. This is a hard
  limit on server memory usage. If the server attempts to allocate memory in excess of this limit, the
  allocation will fail, which may in turn leave the server unable to perform DNS service. Therefore,
  this option is rarely useful as a way of limiting the amount of memory used by the server, but it
  can be used to raise an operating system data size limit that is too small by default. If you wish to
  limit the amount of memory used by the server, use the _max-cache-size_ and _recursive-clients_
  options instead.

files
  The maximum number of files the server may have open concurrently. The default is _unlimited_.

stacksize
  The maximum amount of stack memory the server may use. The default is _default_.

Server Resource Limits
The following options set limits on the server’s resource consumption that are enforced internally by the
server rather than the operating system.

max-ixfr-log-size
  This option is obsolete; it is accepted and ignored for BIND 8 compatibility.

recursive-clients
  The maximum number of simultaneous recursive lookups the server will perform on behalf of
  clients. The default is 1000. Because each recursing client uses a fair bit of memory, on the order
  of 20 kilobytes, the value of the _recursive-clients_ option may have to be decreased on hosts with
  limited memory.
The maximum number of simultaneous client TCP connections that the server will accept. The default is 100.

**max-cache-size**

The maximum amount of memory to use for the server’s cache, in bytes. When the amount of data in the cache reaches this limit, the server will cause records to expire prematurely so that the limit is not exceeded. In a server with multiple views, the limit applies separately to the cache of each view. The default is unlimited, meaning that records are purged from the cache only when their TTLs expire.

**Periodic Task Intervals**

**cleaning-interval**

The server will remove expired resource records from the cache every cleaning-interval minutes. The default is 60 minutes. If set to 0, no periodic cleaning will occur.

**heartbeat-interval**

The server will perform zone maintenance tasks for all zones marked as dialup whenever this interval expires. The default is 60 minutes. Reasonable values are up to 1 day (1440 minutes). If set to 0, no zone maintenance for these zones will occur.

**interface-interval**

The server will scan the network interface list every interface-interval minutes. The default is 60 minutes. If set to 0, interface scanning will only occur when the configuration file is loaded. After the scan, listeners will be started on any new interfaces (provided they are allowed by the listen-on configuration). Listeners on interfaces that have gone away will be cleaned up.

**statistics-interval**

Nameserver statistics will be logged every statistics-interval minutes. The default is 60. If set to 0, no statistics will be logged.

**Note:** Not yet implemented in BIND 9.

**Topology**

All other things being equal, when the server chooses a nameserver to query from a list of nameservers, it prefers the one that is topologically closest to itself. The topology statement takes an address_match_list and interprets it in a special way. Each top-level list element is assigned a distance. Non-negated elements get a distance based on their position in the list, where the closer the match is to the start of the list, the shorter the distance is between it and the server. A negated match will be assigned the maximum distance from the server. If there is no match, the address will get a distance which is further than any non-negated list element, and closer than any negated element. For example,

```plaintext
topology {
  10/8;
  !1.2.3/24;
  { 1.2/16; 3/8; };
};
```

will prefer servers on network 10 the most, followed by hosts on network 1.2.0.0 (netmask 255.255.0.0) and network 3, with the exception of hosts on network 1.2.3 (netmask 255.255.255.0), which is preferred least of all.

The default topology is

```plaintext
topology { localhost; localnets; };
```

**Note:** The topology option is not implemented in BIND 9.
The sortlist Statement

The response to a DNS query may consist of multiple resource records (RRs) forming a resource records set (RRset). The name server will normally return the RRs within the RRset in an indeterminate order (but see the *rrset-order* statement in “RRset Ordering” on page 516). The client resolver code should rearrange the RRs as appropriate, that is, using any addresses on the local net in preference to other addresses. However, not all resolvers can do this or are correctly configured. When a client is using a local server the sorting can be performed in the server, based on the client’s address. This only requires configuring the nameservers, not all the clients.

The sortlist statement (see below) takes an address_match_list and interprets it even more specifically than the topology statement does (“Topology” on page 514). Each top level statement in the sortlist must itself be an explicit address_match_list with one or two elements. The first element (which may be an IP address, an IP prefix, an ACL name or a nested address_match_list) of each top level list is checked against the source address of the query until a match is found.

Once the source address of the query has been matched, if the top level statement contains only one element, the actual primitive element that matched the source address is used to select the address in the response to move to the beginning of the response. If the statement is a list of two elements, then the second element is treated the same as the address_match_list in a topology statement. Each top level element is assigned a distance and the address in the response with the minimum distance is moved to the beginning of the response.

In the following example, any queries received from any of the addresses of the host itself will get responses preferring addresses on any of the locally connected networks. Next most preferred are addresses on the 192.168.1/24 network, and after that either the 192.168.2/24 or 192.168.3/24 network with no preference shown between these two networks. Queries received from a host on the 192.168.1/24 network will prefer other addresses on that network to the 192.168.2/24 and 192.168.3/24 networks. Queries received from a host on the 192.168.4/24 or the 192.168.5/24 network will only prefer other addresses on their directly connected networks.

```
sortlist {
    { localhost; // IF the local host
        { localnets; // THEN first fit on the
            192.168.1/24; // following nets
                { 192.168.2/24; 192.168.3/24; };);
            { 192.168.1/24;
                { 192.168.2/24; 192.168.3/24; };};
            { 192.168.2/24;
                { 192.168.1/24; 192.168.3/24; };};
            { 192.168.3/24;
                { 192.168.1/24; 192.168.2/24; };};
            { 192.168.4/24; 192.168.5/24; }; // if .4 or .5, prefer that net
        };
    };
}
```

The following example will give reasonable behavior for the local host and hosts on directly connected networks. It is similar to the behavior of the address sort in BIND 4.9.x. Responses sent to queries from the local host will favor any of the directly connected networks. Responses sent to queries from any other hosts on a directly connected network will prefer addresses on that same network. Responses to other queries will not be sorted.

```
sortlist {
    { localhost; localnets; };
    { localnets; };
}
```
RRset Ordering
When multiple records are returned in an answer it may be useful to configure the order of the records placed into the response. The `rrset-order` statement permits configuration of the ordering of the records in a multiple record response. See also the `sortlist` statement, "The sortlist Statement" on page 515.

An `order_spec` is defined as follows:

```
[ class class_name ][ type type_name ][ name "domain_name"]
   order ordering
```

If no class is specified, the default is ANY. If no type is specified, the default is ANY. If no name is specified, the default is "*".

The legal values for `ordering` are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fixed</td>
<td>Records are returned in the order they are defined in the zone file.</td>
</tr>
<tr>
<td>random</td>
<td>Records are returned in some random order.</td>
</tr>
<tr>
<td>cyclic</td>
<td>Records are returned in a round-robin order.</td>
</tr>
</tbody>
</table>

For example:

```
rrset-order {
   class IN type A name "host.example.com" order random;
   order cyclic;
};
```

will cause any responses for type A records in class IN that have "host.example.com" as a suffix, to always be returned in random order. All other records are returned in cyclic order.

If multiple `rrset-order` statements appear, they are not combined — the last one applies.

**Note:** The `rrset-order` statement is not yet implemented in BIND 9. BIND 9 currently supports only a "random-cyclic" ordering, where the server randomly chooses a starting point within the RRset and returns the records in order starting at that point, wrapping around the end of the RRset if necessary.

Tuning

**lame-ttl**

Sets the number of seconds to cache a lame server indication. 0 disables caching. (This is not recommended.) Default is 600 (10 minutes). Maximum value is 1800 (30 minutes).

**max-ncache-ttl**

To reduce network traffic and increase performance the server stores negative answers. `max-ncache-ttl` is used to set a maximum retention time for these answers in the server in seconds. The default `max-ncache-ttl` is 10800 seconds (3 hours). `max-ncache-ttl` cannot exceed 7 days and will be silently truncated to 7 days if set to a greater value.

**max-cache-ttl**

`max-cache-ttl` sets the maximum time for which the server will cache ordinary (positive) answers. The default is one week (7 days).

**min-roots**

The minimum number of root servers that is required for a request for the root servers to be accepted. Default is 2.

**Note:** Not yet implemented in BIND 9.

**sig-validity-interval**
Specifies the number of days into the future when DNSSEC signatures automatically generated as a result of dynamic updates ((Section 4.1)) will expire. The default is 30 days. The signature inception time is unconditionally set to one hour before the current time to allow for a limited amount of clock skew.

**min-refresh-time, max-refresh-time, min-retry-time, max-retry-time**

These options control the server’s behavior on refreshing a zone (querying for SOA changes) or retrying failed transfers. Usually the SOA values for the zone are used, but these values are set by the master, giving slave server administrators little control over their contents.

These options allow the administrator to set a minimum and maximum refresh and retry time either per-zone, per-view, or per-server. These options are valid for master, slave and stub zones, and clamp the SOA refresh and retry times to the specified values.

### The Statistics File

The statistics file generated by BIND 9 is similar, but not identical, to that generated by BIND 8.

The statistics dump begins with the line `+++ Statistics Dump +++ (973798949)`, where the number in parentheses is a standard UNIX-style timestamp, measured as seconds since January 1, 1970. Following that line are a series of lines containing a counter type, the value of the counter, optionally a zone name, and optionally a view name. The lines without view and zone listed are global statistics for the entire server. Lines with a zone and view name for the given view and zone (the view name is omitted for the default view). The statistics dump ends with the line `—- Statistics Dump —- (973798949)`, where the number is identical to the number in the beginning line.

The following statistics counters are maintained:

<table>
<thead>
<tr>
<th>Counter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>success</td>
<td>The number of successful queries made to the server or zone. A successful query is defined as query which returns a NOERROR response other than a referral response.</td>
</tr>
<tr>
<td>referral</td>
<td>The number of queries which resulted in referral responses.</td>
</tr>
<tr>
<td>nxrrset</td>
<td>The number of queries which resulted in NOERROR responses with no data.</td>
</tr>
<tr>
<td>nxdomain</td>
<td>The number of queries which resulted in NXDOMAIN responses.</td>
</tr>
<tr>
<td>recursion</td>
<td>The number of queries which caused the server to perform recursion in order to find the final answer.</td>
</tr>
<tr>
<td>failure</td>
<td>The number of queries which resulted in a failure response other than those above.</td>
</tr>
</tbody>
</table>

### server Statement Grammar

```plaintext
server ip_addr {
    [ bogus yes_or_no ; ]
    [ provide-ixfr yes_or_no ; ]
    [ request-ixfr yes_or_no ; ]
    [ edns yes_or_no ; ]
    [ transfers number ; ]
    [ transfer-format ( one-answer | many-answers ) ; ]
    [ keys { string ; [ string ; [ ... ] ] } ; ]
};
```

### server Statement Definition and Usage

The `server` statement defines characteristics to be associated with a remote nameserver.

The `server` statement can occur at the top level of the configuration file or inside a `view` statement. If a `view` statement contains one or more `server` statements, only those apply to the view and any top-level ones are ignored. If a view contains no `server` statements, any top-level `server` statements are used as defaults.
If you discover that a remote server is giving out bad data, marking it as bogus will prevent further queries to it. The default value of **bogus** is **no**.

The **provide-ixfr** clause determines whether the local server, acting as master, will respond with an incremental zone transfer when the given remote server, a slave, requests it. If set to **yes**, incremental transfer will be provided whenever possible. If set to **no**, all transfers to the remote server will be nonincremental. If not set, the value of the **provide-ixfr** option in the view or global options block is used as a default.

The **request-ixfr** clause determines whether the local server, acting as a slave, will request incremental zone transfers from the given remote server, a master. If not set, the value of the **request-ixfr** option in the view or global options block is used as a default.

IXFR requests to servers that do not support IXFR will automatically fall back to AXFR. Therefore, there is no need to manually list which servers support IXFR and which ones do not; the global default of **yes** should always work. The purpose of the **provide-ixfr** and **request-ixfr** clauses is to make it possible to disable the use of IXFR even when both master and slave claim to support it, for example if one of the servers is buggy and crashes or corrupts data when IXFR is used.

The **edns** clause determines whether the local server will attempt to use EDNS when communicating with the remote server. The default is **yes**.

The server supports two zone transfer methods. The first, **one-answer**, uses one DNS message per resource record transferred. **many-answers** packs as many resource records as possible into a message. **many-answers** is more efficient, but is only known to be understood by BIND 9, BIND 8.x, and patched versions of BIND 4.9.5. You can specify which method to use for a server with the **transfer-format** option. If **transfer-format** is not specified, the **transfer-format** specified by the **options** statement will be used.

**transfers** is used to limit the number of concurrent inbound zone transfers from the specified server. If no **transfers** clause is specified, the limit is set according to the **transfers-per-ns** option.

The **keys** clause is used to identify a **key_id** defined by the **key** statement, to be used for transaction security when talking to the remote server. The **key** statement must come before the **server** statement that references it. When a request is sent to the remote server, a request signature will be generated using the key specified here and appended to the message. A request originating from the remote server is not required to be signed by this key.

Although the grammar of the **keys** clause allows for multiple keys, only a single key per server is currently supported.

**trusted-keys Statement Grammar**

```plaintext
trusted-keys {
    string number number number string;
    [ string number number number string; [...] ]
};
```

**trusted-keys Statement Definition and Usage**

The **trusted-keys** statement defines DNSSEC security roots. A security root is defined when the public key for a non-authoritative zone is known, but cannot be securely obtained through DNS, either because it is the DNS root zone or its parent zone is unsigned. Once a key has been configured as a trusted key, it is treated as if it had been validated and proven secure. The resolver attempts DNSSEC validation on all DNS data in subdomains of a security root.

The **trusted-keys** statement can contain multiple key entries, each consisting of the key’s domain name, flags, protocol, algorithm, and the base-64 representation of the key data.
6.2.20. view Statement Definition and Usage

The view statement is a powerful new feature of BIND 9 that lets a name server answer a DNS query differently depending on who is asking. It is particularly useful for implementing split DNS setups without having to run multiple servers.

Each view statement defines a view of the DNS namespace that will be seen by a subset of clients. A client matches a view if its source IP address matches the address_match_list of the view's match-clients clause and its destination IP address matches the address_match_list of the view's match-destinations clause. If not specified, both match-clients and match-destinations default to matching all addresses. A view can also be specified as match-recursive-only, which means that only recursive requests from matching clients will match that view. The order of the view statements is significant — a client request will be resolved in the context of the first view that it matches.

Zones defined within a view statement will be only be accessible to clients that match the view. By defining a zone of the same name in multiple views, different zone data can be given to different clients, for example, “internal” and “external” clients in a split DNS setup.

Many of the options given in the options statement can also be used within a view statement, and then apply only when resolving queries with that view. When no view-specific value is given, the value in the options statement is used as a default. Also, zone options can have default values specified in the view statement; these view-specific defaults take precedence over those in the options statement.

Views are class specific. If no class is given, class IN is assumed. Note that all non-IN views must contain a hint zone, since only the IN class has compiled-in default hints.

If there are no view statements in the config file, a default view that matches any client is automatically created in class IN, and any zone statements specified on the top level of the configuration file are considered to be part of this default view. If any explicit view statements are present, all zone statements must occur inside view statements.

Here is an example of a typical split DNS setup implemented using view statements.

```plaintext
view "internal" {
    // This should match our internal networks.
    match-clients { 10.0.0.0/8; };
    // Provide recursive service to internal clients only.
    recursion yes;
    // Provide a complete view of the example.com zone
    // including addresses of internal hosts.
    zone "example.com" {
        type master;
        file "example-internal.db";
    };
};
view "external" {
    match-clients { any; };
    // Refuse recursive service to external clients.
    recursion no;
    // Provide a restricted view of the example.com zone
    // containing only publicly accessible hosts.
    zone "example.com" {
```

Chapter 2. File Formats 519
type master;
file "example-external.db";
};

zone Statement Grammar

zone zone_name [class] []
  type ( master | slave | hint | stub | forward ) ;
  [ allow-notify { address_match_list ] ; ]
  [ allow-query { address_match_list ] ; ]
  [ allow-transfer { address_match_list ] ; ]
  [ allow-update { address_match_list ] ; ]
  [ update-policy { update_policy_rule ] ; ]
  [ allow-update-forwarding { address_match_list ] ; ]
  [ also-notify { address_match_list ] ; ]
  [ also-notify { address_match_list ] ; ]
  [ check-names { warn|fail|ignore ] ; ]
  [ dialup dialup_option ] ;
  [ file string ] ;
  [ forward (only|first) ] ;
  [ forwarders { ip_addr [port ip_port] ; [ ip_addr [port ip_port] ; ... ] } ; ]
  [ ixfr-base string ] ;
  [ ixfr-tmp-file string ] ;
  [ maintain-ixfr-base yes_or_no ] ;
  [ masters [port ip_port] [ ip_addr [port ip_port] [key key] ; [ ... ] ] ; ]
  [ max-ixfr-log-size number ] ;
  [ max-transfer-idle-in number ] ;
  [ max-transfer-idle-out number ] ;
  [ max-transfer-time-in number ] ;
  [ max-transfer-time-out number ] ;
  [ notify yes_or_no | explicit ] ;
  [ pubkey number number number number string ] ;
  [ transfer-source (ip4_addr | *) [port ip_port] ] ;
  [ transfer-source-v6 (ip6_addr | *) [port ip_port] ] ;
  [ notify-source (ip4_addr | *) [port ip_port] ] ;
  [ notify-source-v6 (ip6_addr | *) [port ip_port] ] ;
  [ zone-statistics yes_or_no ] ;
  [ sig-validity-interval number ] ;
  [ database string ] ;
  [ min-refresh-time number ] ;
  [ max-refresh-time number ] ;
  [ min-retry-time number ] ;
  [ max-retry-time number ] ;
}]

zone Statement Definition and Usage

Zone Types:

master The server has a master copy of the data for the zone and will be able to provide authoritative answers for it.

slave A slave zone is a replica of a master zone. The masters list specifies one or more IP addresses of master servers that the slave contacts to update its copy of the zone. By default, transfers are made from port 53 on the servers; this can be changed for all servers by specifying a port number before the list of IP addresses, or on a per-server basis after the IP address. Authentication to the master can also be done with per-server TSIG keys. If a file is specified, then the replica will be written to this file whenever the zone is changed, and reloaded from this file on a server restart. Use of a file is recommended, since it often speeds server start-up and eliminates a needless waste of bandwidth. Note that for large numbers (in the tens or hundreds of thousands) of zones per server, it is best to use a two level naming scheme for zone file names. For example, a slave server for the zone example.com might place the zone contents into a file called ex/example.com where ex/ is just the first two letters of the zone name. (Most operating systems behave very slowly if you put 100K files into a single directory.)
A stub zone is similar to a slave zone, except that it replicates only the NS records of a master zone instead of the entire zone. Stub zones are not a standard part of the DNS; they are a feature specific to the BIND implementation.

Stub zones can be used to eliminate the need for glue NS record in a parent zone at the expense of maintaining a stub zone entry and a set of name server addresses in named.conf. This usage is not recommended for new configurations, and BIND 9 supports it only in a limited way. In BIND 4/8, zone transfers of a parent zone included the NS records from stub children of that zone. This meant that, in some cases, users could get away with configuring child stubs only in the master server for the parent zone. BIND 9 never mixes together zone data from different zones in this way. Therefore, if a BIND 9 master serving a parent zone has child stub zones configured, all the slave servers for the parent zone also need to have the same child stub zones configured.

Stub zones can also be used as a way of forcing the resolution of a given domain to use a particular set of authoritative servers. For example, the caching name servers on a private network using RFC2157 addressing may be configured with stub zones for 10.in-addr.arpa to use a set of internal name servers as the authoritative servers for that domain.

A "forward zone" is a way to configure forwarding on a per-domain basis. A zone statement of type forward can contain a forward and/or forwarders statement, which will apply to queries within the domain given by the zone name. If no forwarders statement is present or an empty list for forwarders is given, then no forwarding will be done for the domain, canceling the effects of any forwarders in the options statement. Thus if you want to use this type of zone to change the behavior of the global forward option (that is, "forward first to", then "forward only", or vice versa, but want to use the same servers as set globally) you need to respecify the global forwarders.

The initial set of root nameservers is specified using a "hint zone". When the server starts up, it uses the root hints to find a root nameserver and get the most recent list of root nameservers. If no hint zone is specified for class IN, the server uses a compiled-in default set of root servers hints. Classes other than IN have no built-in defaults hints.

**Class:** The zone's name may optionally be followed by a class. If a class is not specified, class IN (for Internet), is assumed. This is correct for the vast majority of cases.

The hesiod class is named for an information service from MIT's Project Athena. It is used to share information about various systems databases, such as users, groups, printers and so on. The keyword HS is a synonym for hesiod.

Another MIT development is CHAOSnet, a LAN protocol created in the mid-1970s. Zone data for it can be specified with the CHAOS class.

**Zone Options:**

allow-notify

See the description of allow-notify in "Access Control" on page 511.

allow-query

See the description of allow-query in "Access Control" on page 511.

allow-transfer

See the description of allow-transfer in "Access Control" on page 511.

allow-update

Specifies which hosts are allowed to submit Dynamic DNS updates for master zones. The default is to deny updates from all hosts.

update-policy
Specifies a "Simple Secure Update" policy. See "Dynamic Update Policies" on page 523.

allow-update-forwarding

Specifies which hosts are allowed to submit Dynamic DNS updates to slave zones to be forwarded to the master. The default is { none; }, which means that no update forwarding will be performed. To enable update forwarding, specify allow-update-forwarding { any; };. Specifying values other than { none; } or { any; } is usually counterproductive, since the responsibility for update access control should rest with the master server, not the slaves.

Note that enabling the update forwarding feature on a slave server may expose master servers relying on insecure IP address based access control to attacks.

database

Specify the type of database to be used for storing the zone data. The string following the database keyword is interpreted as a list of whitespace-delimited words. The first word identifies the database type, and any subsequent words are passed as arguments to the database to be interpreted in a way specific to the database type.

The default is "rbt", BIND 9's native in-memory red-black-tree database. This database does not take arguments.

Other values are possible if additional database drivers have been linked into the server. Some sample drivers are included with the distribution but none are linked in by default.
max-transfer-idle-in

See the description of max-transfer-idle-in in “Zone Transfers” on page 511.

max-transfer-time-out

See the description of max-transfer-time-out in “Zone Transfers” on page 511.

max-transfer-idle-out

See the description of max-transfer-idle-out in “Zone Transfers” on page 511.

notify

See the description of notify in “Boolean Options” on page 508.

pubkey

In BIND 8, this option was intended for specifying a public zone key for verification of signatures in DNSSEC signed zones when they are loaded from disk. BIND 9 does not verify signatures on loading and ignores the option.

zone-statistics

If yes, the server will keep statistical information for this zone, which can be dumped to the statistics-file defined in the server options.

sig-validity-interval

See the description of sig-validity-interval in “Tuning” on page 516.

transfer-source

See the description of transfer-source in “Zone Transfers” on page 511

notify-source

See the description of notify-source in “Zone Transfers” on page 511

min-refresh-time, max-refresh-time, min-retry-time, max-retry-time

See the description in “Tuning” on page 516.

Dynamic Update Policies: BIND 9 supports two alternative methods of granting clients the right to perform dynamic updates to a zone, configured by the allow-update and update-policy option, respectively.

The allow-update clause works the same way as in previous versions of BIND. It grants given clients the permission to update any record of any name in the zone.

The update-policy clause is new in BIND 9 and allows more fine-grained control over what updates are allowed. A set of rules is specified, where each rule either grants or denies permissions for one or more names to be updated by one or more identities. If the dynamic update request message is signed (that is, it includes either a TSIG or SIG(0) record), the identity of the signer can be determined.

Rules are specified in the update-policy zone option, and are only meaningful for master zones. When the update-policy statement is present, it is a configuration error for the allow-update statement to be present. The update-policy statement only examines the signer of a message; the source address is not relevant.

This is how a rule definition looks:

( grant | deny ) identity nametype name [ types ]
Each rule grants or denies privileges. Once a message has successfully matched a rule, the operation is immediately granted or denied and no further rules are examined. A rule is matched when the signer matches the identity field, the name matches the name field, and the type is specified in the type field.

The identity field specifies a name or a wildcard name. The nametype field has 4 values: name, subdomain, wildcard, and self.

<table>
<thead>
<tr>
<th>nametype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Matches when the updated name is the same as the name in the name field.</td>
</tr>
<tr>
<td>subdomain</td>
<td>Matches when the updated name is a subdomain of the name in the name field (which includes the name itself).</td>
</tr>
<tr>
<td>wildcard</td>
<td>Matches when the updated name is a valid expansion of the wildcard name in the name field.</td>
</tr>
<tr>
<td>self</td>
<td>Matches when the updated name is the same as the message signer. The name field is ignored.</td>
</tr>
</tbody>
</table>

If no types are specified, the rule matches all types except SIG, NS, SOA, and NXT. Types may be specified by name, including "ANY" (ANY matches all types except NXT, which can never be updated).

**Zone File:**

*DOMAIN Data File, DOMAIN Reverse Data File, DOMAIN Cache File, and DOMAIN Local: Types of Resource Records and When to Use Them*

This section, largely borrowed from RFC 1034, describes the concept of a Resource Record (RR) and explains when each is used. Since the publication of RFC 1034, several new RRs have been identified and implemented in the DNS. These are also included.

**Resource Records**

A domain name identifies a node. Each node has a set of resource information, which may be empty. The set of resource information associated with a particular name is composed of separate RRs. The order of RRs in a set is not significant and need not be preserved by nameservers, resolvers, or other parts of the DNS. However, sorting of multiple RRs is permitted for optimization purposes, for example, to specify that a particular nearby server be tried first. See "The sortlist Statement" on page 515 and "RRset Ordering" on page 516.

The components of a Resource Record are:

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>owner name</td>
<td>the domain name where the RR is found.</td>
</tr>
<tr>
<td>type</td>
<td>an encoded 16 bit value that specifies the type of the resource in this resource record. Types refer to abstract resources.</td>
</tr>
<tr>
<td>TTL</td>
<td>the time to live of the RR. This field is a 32 bit integer in units of seconds, and is primarily used by resolvers when they cache RRs. The TTL describes how long a RR can be cached before it should be discarded.</td>
</tr>
<tr>
<td>class</td>
<td>an encoded 16 bit value that identifies a protocol family or instance of a protocol.</td>
</tr>
<tr>
<td>RDATA</td>
<td>the type and sometimes class-dependent data that describes the resource.</td>
</tr>
</tbody>
</table>

The following are types of valid RRs (some of these listed, although not obsolete, are experimental (x) or historical (h) and no longer in general use):

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>a host address.</td>
</tr>
<tr>
<td>A6</td>
<td>an IPv6 address.</td>
</tr>
<tr>
<td>AAAA</td>
<td>Obsolete format of IPv6 address</td>
</tr>
</tbody>
</table>
The following classes of resource records are currently valid in the DNS:

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>the Internet system.</td>
</tr>
</tbody>
</table>

For information about other, older classes of RRs

**RDATA** is the type-dependent or class-dependent data that describes the resource:

<table>
<thead>
<tr>
<th>RDATA Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>for the IN class, a 32 bit IP address.</td>
</tr>
<tr>
<td>A6</td>
<td>maps a domain name to an IPv6 address, with a provision for indirection for leading &quot;prefix&quot; bits.</td>
</tr>
<tr>
<td>CNAME</td>
<td>a domain name.</td>
</tr>
<tr>
<td>DNAME</td>
<td>provides alternate naming to an entire subtree of the domain name space, rather than to a single node. It causes some suffix of a queried name to be substituted with a name from the DNAME record's RDATA.</td>
</tr>
<tr>
<td>MX</td>
<td>a 16 bit preference value (lower is better) followed by a host name willing to act as a mail exchange for the owner domain.</td>
</tr>
<tr>
<td>NS</td>
<td>a fully qualified domain name.</td>
</tr>
<tr>
<td>PTR</td>
<td>a fully qualified domain name.</td>
</tr>
<tr>
<td>SOA</td>
<td>several fields.</td>
</tr>
</tbody>
</table>

The owner name is often implicit, rather than forming an integral part of the RR. For example, many nameservers internally form tree or hash structures for the name space, and chain RRs off nodes. The
remaining RR parts are the fixed header (type, class, TTL) which is consistent for all RRs, and a variable part (RDATA) that fits the needs of the resource being described.

The meaning of the TTL field is a time limit on how long an RR can be kept in a cache. This limit does not apply to authoritative data in zones; it is also timed out, but by the refreshing policies for the zone. The TTL is assigned by the administrator for the zone where the data originates. While short TTLs can be used to minimize caching, and a zero TTL prohibits caching, the realities of Internet performance suggest that these times should be on the order of days for the typical host. If a change can be anticipated, the TTL can be reduced prior to the change to minimize inconsistency during the change, and then increased back to its former value following the change.

The data in the RDATA section of RRs is carried as a combination of binary strings and domain names. The domain names are frequently used as “pointers” to other data in the DNS.

Textual expression of RRs

RRs are represented in binary form in the packets of the DNS protocol, and are usually represented in highly encoded form when stored in a nameserver or resolver. In the examples provided in RFC 1034, a style similar to that used in master files was employed in order to show the contents of RRs. In this format, most RRs are shown on a single line, although continuation lines are possible using parentheses.

The start of the line gives the owner of the RR. If a line begins with a blank, then the owner is assumed to be the same as that of the previous RR. Blank lines are often included for readability.

Following the owner, we list the TTL, type, and class of the RR. Class and type use the mnemonics defined above, and TTL is an integer before the type field. In order to avoid ambiguity in parsing, type and class mnemonics are disjoint, TTLs are integers, and the type mnemonic is always last. The IN class and TTL values are often omitted from examples in the interests of clarity.

The resource data or RDATA section of the RR are given using knowledge of the typical representation for the data.

For example, we might show the RRs carried in a message as:

<table>
<thead>
<tr>
<th>Owner</th>
<th>Type</th>
<th>RDATA</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISI.EDU.</td>
<td>MX</td>
<td>10 VENERA.ISI.EDU.</td>
</tr>
<tr>
<td></td>
<td>MX</td>
<td>10 VAXA.ISI.EDU</td>
</tr>
<tr>
<td>VENERA.ISI.EDU</td>
<td>A</td>
<td>128.9.0.32</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>10.1.0.52</td>
</tr>
<tr>
<td>VAXA.ISI.EDU</td>
<td>A</td>
<td>10.2.0.27</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>128.9.0.33</td>
</tr>
</tbody>
</table>

The MX RRs have an RDATA section which consists of a 16 bit number followed by a domain name. The address RRs use a standard IP address format to contain a 32 bit internet address.

This example shows six RRs, with two RRs at each of three domain names.

Similarly we might see:

<table>
<thead>
<tr>
<th>Owner</th>
<th>Class</th>
<th>RDATA</th>
</tr>
</thead>
<tbody>
<tr>
<td>XX.LCS.MIT.EDU.</td>
<td>A</td>
<td>10.0.0.44</td>
</tr>
<tr>
<td>CH</td>
<td>A</td>
<td>MIT.EDU. 2420</td>
</tr>
</tbody>
</table>

This example shows two addresses for XX.LCS.MIT.EDU, each of a different class.
Discussion of MX Records

As described above, domain servers store information as a series of resource records, each of which contains a particular piece of information about a given domain name (which is usually, but not always, a host). The simplest way to think of a RR is as a typed pair of datum, a domain name matched with relevant data, and stored with some additional type information to help systems determine when the RR is relevant.

MX records are used to control delivery of email. The data specified in the record is a priority and a domain name. The priority controls the order in which email delivery is attempted, with the lowest number first. If two priorities are the same, a server is chosen randomly. If no servers at a given priority are responding, the mail transport agent will fall back to the next largest priority. Priority numbers do not have any absolute meaning — they are relevant only respective to other MX records for that domain name. The domain name given is the machine to which the mail will be delivered. It must have an associated A record — CNAME is not sufficient.

For a given domain, if there is both a CNAME record and an MX record, the MX record is in error, and will be ignored. Instead, the mail will be delivered to the server specified in the MX record pointed to by the CNAME.

example.com. IN MX 10 mail.example.com.
IN MX 10 mail2.example.com.
IN MX 10 mail.backup.org.
mail.example.com. IN A 10.0.0.1
mail2.example.com. IN A 10.0.0.2

For example:

Mail delivery will be attempted to mail.example.com and mail2.example.com (in any order), and if neither of those succeed, delivery to mail.backup.org will be attempted.

Setting TTLs

The time to live of the RR field is a 32 bit integer represented in units of seconds, and is primarily used by resolvers when they cache RRs. The TTL describes how long a RR can be cached before it should be discarded. The following three types of TTL are currently used in a zone file.

<table>
<thead>
<tr>
<th>SOA</th>
<th>The last field in the SOA is the negative caching TTL. This controls how long other servers will cache no-such-domain (NXDOMAIN) responses from you. The maximum time for negative caching is 3 hours (3h).</th>
</tr>
</thead>
<tbody>
<tr>
<td>$TTL</td>
<td>The $TTL directive at the top of the zone file (before the SOA) gives a default TTL for every RR without a specific TTL set.</td>
</tr>
<tr>
<td>RR TTLs</td>
<td>Each RR can have a TTL as the second field in the RR, which will control how long other servers can cache the it.</td>
</tr>
</tbody>
</table>

All of these TTLs default to units of seconds, though units can be explicitly specified, for example, 1h30m.

Inverse Mapping in IPv4

Reverse name resolution (that is, translation from IP address to name) is achieved by means of the in-addr.arpa domain and PTR records. Entries in the in-addr.arpa domain are made in least-to-most significant order, read left to right. This is the opposite order to the way IP addresses are usually written. Thus, a machine with an IP address of 10.1.2.3 would have a corresponding in-addr.arpa name of 3.2.1.10.in-addr.arpa. This name should have a PTR resource record whose data field is the name of the machine or, optionally, multiple PTR records if the machine has more than one name. For example, in the [ example.com ] domain:
Note: The $ORIGIN lines in the examples are for providing context to the examples only—they do not necessarily appear in the actual usage. They are only used here to indicate that the example is relative to the listed origin.

Other Zone File Directives

The Master File Format was initially defined in RFC 1035 and has subsequently been extended. While the Master File Format itself is class independent all records in a Master File must be of the same class.

Master File Directives include $ORIGIN, $INCLUDEx and $TTL.

The $ORIGIN Directive

Syntax: $ORIGIN domain-name [ comment ]

$ORIGIN sets the domain name that will be appended to any unqualified records. When a zone is first read in there is an implicit $ORIGIN < zone-name > . The current $ORIGIN is appended to the domain specified in the $ORIGIN argument if it is not absolute.

$ORIGIN example.com.
WWW CNAME MAIN-SERVER

is equivalent to
WWW.EXAMPLE.COM. CNAME MAIN-SERVER.EXAMPLE.COM.

The SINCLUDE Directive

Syntax: SINCLUDE filename [ origin ] [ comment ]

Read and process the file filename as if it were included into the file at this point. If origin is specified the file is processed with $ORIGIN set to that value, otherwise the current $ORIGIN is used.

The origin and the current domain name revert to the values they had prior to the SINCLUDE once the file has been read.

Note: RFC 1035 specifies that the current origin should be restored after an SINCLUDE, but it is silent on whether the current domain name should also be restored. BIND 9 restores both of them. This could be construed as a deviation from RFC 1035, a feature, or both.

The $TTL Directive

Syntax: $TTL default-ttl [ comment ]

Set the default Time To Live (TTL) for subsequent records with undefined TTLs. Valid TTLs are of the range 0-2147483647 seconds.

$TTL is defined in RFC 2308.

BIND Master File Extension: the $GENERATE Directive

Syntax: $GENERATE range lhs type rhs [ comment ]
$GENERATE is used to create a series of resource records that only differ from each other by an iterator.
$GENERATE can be used to easily generate the sets of records required to support sub /24 reverse
degolutions described in RFC 2317: Classless IN-ADDR.ARPA delegation.

$ORIGIN 0.0.192.IN-ADDR.ARPA.
$GENERATE 1-2 0 NS SERVER$.EXAMPLE.
$GENERATE 1-127 $ CNAME $.0

is equivalent to
0.0.0.192.IN-ADDR.ARPA NS SERVER1.EXAMPLE.
0.0.0.192.IN-ADDR.ARPA NS SERVER2.EXAMPLE.
1.0.0.192.IN-ADDR.ARPA CNAME 1.0.0.0.192.IN-ADDR.ARPA
2.0.0.192.IN-ADDR.ARPA CNAME 2.0.0.0.192.IN-ADDR.ARPA
...
127.0.0.192.IN-ADDR.ARPA CNAME 127.0.0.0.192.IN-ADDR.ARPA

range This can be one of two forms: start-stop or start-stop/step. If the first form is used then step is
set to 1. All of start, stop and step must be positive.

lhs lhs describes the owner name of the resource records to be created. Any single $ symbols
within the lhs side are replaced by the iterator value. To get a $ in the output you need to
escape the $ using a backslash \, e.g. \\$. The $ may optionally be followed by modifiers which
change the offset from the interator, field width and base. Modifiers are introduced by a { immediately following the $ as $[offset[,width[,base]]]. e.g. $[-20,3,d] which subtracts 20 from
the current value, prints the result as a decimal in a zero padded field of with 3. Available output
forms are decimal (d), octal (o) and hexadecimal (x or X for uppercase). The default modifier is
$[0,0,d]. If the lhs is not absolute, the current $ORIGIN is appended to the name.

For compatibility with earlier versions $$ is still recognised a indicating a literal $ in the output.

type At present the only supported types are PTR, CNAME, DNAME, A, AAAA and NS.

rhs rhs is a domain name. It is processed similarly to lhs.

The $GENERATE directive is a BIND extension and not part of the standard zone file format.

Files
/usr/samples/tcpip/named.conf Contains the sample named.conf file.

Related Information
The named daemon.

The syslogd daemon.

The DOMAIN cache file format, DOMAIN local file format, DOMAIN data file format, DOMAIN Reverse
data file format, rc.tcpip file format.

Configuring domain name servers and TCP/IP name resolution in Networks and communication
management.

.netrc File Format for TCP/IP

Purpose
Specifies automatic login information for the ftp and rexec commands.
Description
The $HOME/.netrc file contains information used by the automatic login feature of the `rexeC` and `ftp` commands. It is a hidden file in a user’s home directory and must be owned either by the user executing the command or by the root user. If the .netrc file contains a login password, the file’s permissions must be set to 600 (read and write by owner only). This file is part of TCP/IP in Network Support Facilities.

Note: The .netrc file is not used by any programs when the `securetcpip` command is running on your system.

The .netrc can contain the following entries (separated by spaces, tabs, or new lines):

- **machine HostName**
  - The HostName variable is the name of a remote host. This entry begins the definition of the automatic login process for the specified host. All following entries up to the next machine entry or the end of the file apply to that host.

- **default**
  - The default variable is the same as machine except that default matches any name. There can be only one default entry. It must be the last entry (after all machine entries); otherwise, entries that follow it will be ignored. This is normally used as:
    ```
    default login anonymous password user@site
    ```
    thereby giving the user automatic anonymous ftp login to machines not specified in the .netrc file. This can be overridden by using the -n flag to disable the auto-login.

- **login UserName**
  - The UserName variable is the full domain user name for use at the remote host. If this entry is found, the automatic login process initiates a login, using the specified name. If this entry is missing, the automatic login process is unsuccessful.

- **password Password**
  - The Password variable is the login password to be used. The automatic login process supplies this password to the remote server. A login password must be established at the remote host, and that password must be entered in the .netrc file. Otherwise the automatic login process is unsuccessful, and the user is prompted for the login password.

- **account Password**
  - The Password variable is the account password to be used. If this entry is found and an account password is required at the remote host, the automatic login process supplies the password to the remote server. If the remote host requires an account password but this entry is missing, the automatic login process prompts for the account password.

- **macdef MacroName**
  - The MacroName variable is the name of an ftp subcommand macro. The macro is defined to contain all of the following ftp subcommands up to the next blank line or the end of the file. If the macro is named init, the ftp command executes the macro upon successful completion of the automatic login process. The `rexeC` command does not recognize a macdef entry.

Examples
The following is an example of an entry in a .netrc file:

```
machine host1.austin.century.com login fred password bluebonnet
```

Files

```
/usr/samples/tcpip/netrc
```
Sample .netrc file

Related Information

The `ftp` command, `rexeC` command, `securetcpip` command. Creating the .netrc file in *Networks and communication management.*
networks File Format for TCP/IP

**Purpose**
Contains network name information.

**Description**
The `/etc/networks` file contains information about the known networks that comprise the DARPA Internet. Each network is represented by a single line in the `networks` file. The format for the entries in the `networks` file is:

*Name* *Number* *Aliases*

The fields are described as follows:

- **Name**: Specifies an official network name.
- **Number**: Specifies a network number.
- **Aliases**: Specifies any unofficial names used for the network.

Items on a line are separated by one or more spaces or tab characters. Comments begin with a # (pound sign). Routines that search the `networks` file do not interpret characters from the beginning of a comment to the end of that line. Network numbers are specified in dotted-decimal notation. A network name can contain any printable character except a field delimiter, new-line character, or comment character.

The `networks` file is normally created from the official network database maintained at the Network Information Center (NIC). The file can be modified locally to include unofficial aliases or unknown networks. This file is part of TCP/IP in Network Support Facilities.

**Files**

`/usr/lpp/tcpip/samples/networks`  
Contains a sample `networks` file, which also contains directions for its use.

**Related Information**
The `routed` daemon.
The `getnetent` subroutine.

TCP/IP name resolution in *Networks and communication management*.

nroff or troff Input File Format

**Purpose**
Specifies input file format for the `nroff` and `troff` commands.

**Description**
The `nroff` and `troff` commands format text for printing by interspersing the text with control sequences. Control sequences are either control line requests or escape requests that control text processing by the printing device.

Control lines begin with a control character followed by a one- or two-character name that specifies a basic request or a user-defined macro. Default control characters are the . (dot) or the ’ (apostrophe). The
' (apostrophe) control character suppresses the nroff or troff command break function, which is caused by some requests. This break function forces output of a partially filled line. To separate the control character from the request or macro, use white space created with either a tab or the space bar. The nroff and troff commands ignore control lines with unrecognized names.

Escape requests can be inserted anywhere in the input text by means of an escape character. The \ (backslash) character is the default escape character. For example, the escape request \nr causes the contents of the number register, r, to be read.

Note: If text must begin a line with a . (dot), a zero-width character sequence (\&) must precede the control character. This is true even if the control character is preceded by an escape request. The zero-width character prevents the command from interpreting the text as a control character. See the example for an illustration of the use of a zero-width character.

Examples
To print the words .dean, enter:
\fB\&.dean

If you neglected to add the \&, the formatter would read the statement as the macro request:
.dean

Related Information
The nroff command, troff command.

The nroff and troff Requests for the nroff and troff Commands

nterm File Format

Purpose
Describes terminal driving tables for the nroff command.

Description
The nroff command uses driving tables to customize its output for various types of output devices such as printing terminals, special word-processing terminals (such as Diablo, Qume, or NEC Spinwriter mechanisms), or special output-filter programs. These driving tables are written as ASCII files and are installed in the /usr/share/lib/nterm/tab.Name file, where the Name variable is the name for a terminal type.

The first line of a driving table should contain the name of the terminal, which is simply a string with no imbedded white space (any combination of spaces, tabs, and newline characters). The next part of the driver table is structured as follows:

- bset [Integer]
- breset [Integer]
- hor [Integer]
- vert [Integer]
- newline [Integer]
- char [Integer]
- em [Integer]
- halfline [Integer]
- adj [Integer]
• **twinit** [Character String]
• **twrest** [Character String]
• **twnl** [Character String]
• **hlr** [Character String]
• **hlf** [Character String]
• **flr** [Character String]
• **bdon** [Character String]
• **bdoff** [Character String]
• **iton** [Character String]
• **itoff** [Character String]
• **ploton** [Character String]
• **plotoff** [Character String]
• **up** [Character String]
• **down** [Character String]
• **right** [Character String]
• **left** [Character String]
• **codeset** [Character String]

The meanings of these fields are as follows:

**bset**
- Specifies bits to set in the **c_oflag** field of the *termio* structure before output.

**breset**
- Specifies bits to reset in the **c_oflag** field of the *termio* structure before output.

**hor**
- Specifies horizontal resolution in units of 1/240 of an inch.

**vert**
- Defines vertical resolution in units of 1/240 of an inch.

**newline**
- Defines space moved by a new-line (linefeed) character in units of 1/240 of an inch.

**char**
- Defines a quantum of character sizes, in units of 1/240 of an inch (that is, a character is a multiple of **char** units wide).

**em**
- Defines the size of an em space in units of 1/240 of an inch.

**halfline**
- Defines the amount of space moved by a half-linefeed (or half-reverse-linefeed) character in units of 1/240 of an inch.

**adj**
- Defines a quantum of white space, in 1/240 of an inch; that is, white spaces are a multiple of **adj** units wide.

**Note:** If this is less than the size of the space character, the *nroff* command outputs fractional spaces using plot mode. Also, if the *-e* switch to the *nroff* command is used, the **adj** variable is set equal to the **hor** variable by the *nroff* command.

**twinit**
- Specifies a sequence of characters used to initialize the terminal in a mode suitable for the *nroff* command.

**twrest**
- Specifies a sequence of characters used to restore the terminal to normal mode.

**twnl**
- Specifies a sequence of characters used to move down one line.

**hlr**
- Specifies a sequence of characters used to move up one-half line.

**hlf**
- Specifies a sequence of characters used to move down one-half line.

**flr**
- Specifies a sequence of characters used to move up one line.

**bdon**
- Specifies a sequence of characters used to turn on hardware boldface mode, if any.

**bdoff**
- Specifies a sequence of characters used to turn off hardware boldface mode, if any.

**iton**
- Specifies a sequence of characters used to turn on hardware italics mode, if any.

**itoff**
- Specifies a sequence of characters used to turn off hardware italics mode, if any.

**ploton**
- Specifies a sequence of characters used to turn on hardware plot mode (for Diablo-type mechanisms), if any.

**plotoff**
- Specifies a sequence of characters used to turn off hardware plot mode (for Diablo-type mechanisms), if any.

**up**
- Specifies a sequence of characters used to move up one resolution unit (**vert**) in plot mode, if any.
down Specifies a sequence of characters used to move down one resolution unit (vert) in plot mode, if any.

right Specifies a sequence of characters used to move right one resolution unit (hor) in plot mode, if any.

left Specifies a sequence of characters used to move left one resolution unit (hor) in plot mode, if any.

codeset

CodeSetName Specifies the code set for the particular output device. CodesetName is any valid name for use with the iconv command. The code set defines character entries within the font description file for the character set section. The code set field is optional. If used, the code set field must follow the “left” field and precede the character set section, if provided. The default is IBM-850.

The nroff command uses the specified CodesetName and the code set implied by the current locale to determine if code set conversions are necessary for the input characters. The iconv function is used to perform the code set conversion if necessary.

This part of the driving table is fixed-format; you cannot change the order of entries. Entries should be on separate lines each containing two fields (no comments allowed) separated by white space; for example:

| bset  | 0 |
| breset | 0 |
| Hor  | 24 |

Follow this first part of the driving table with a line containing only the word charset, and then specify a table of special characters that you want to include. That is, specify all the non-ASCII characters that the nroff command knows by 2-character names, such as \(hy. If the nroff command does not find the word charset where it expects, it terminates processing with an error message.

Each definition after charset occupies one line and has the following format:

chname width output

The chname field is the (2-letter) name of the special character, the width field is its width in ems, and the output field is the string of characters and escape sequences to send to the terminal to produce the special character.

International Character Support

For fonts for large character sets in which most characters are the same width, as in Japanese, Chinese, and Korean, prototype characters are provided for the character set section of the nterm table. These prototype characters specify the width of characters of varying byte lengths. The code field for prototype character entries must contain a single ? (question mark). The prototype character entries apply to all characters not explicitly defined on their own in the character set section. It is assumed the output device code for characters handled via prototype characters is the same as the input code for characters (with possible codeset conversions). The following are the prototype character definitions:

<table>
<thead>
<tr>
<th>Width</th>
<th>Represents the width of all one-byte characters not defined elsewhere.</th>
</tr>
</thead>
<tbody>
<tr>
<td>X1</td>
<td>Width ? Represents the width of all one-byte characters not defined elsewhere.</td>
</tr>
<tr>
<td>X2</td>
<td>Width ? Represents the width of all two-byte characters not defined elsewhere.</td>
</tr>
<tr>
<td>X3</td>
<td>Width ? Represents the width of all three-byte characters not defined elsewhere.</td>
</tr>
<tr>
<td>X4</td>
<td>Width ? Represents the width of all four-byte characters not defined elsewhere.</td>
</tr>
</tbody>
</table>

If any field in the charset part of the driving table does not pertain to the output device, you can give that particular sequence as a null string or leave out the entry. Special characters that do not have a definition in this file are ignored on output by the nroff command.

You can put the charset definitions in any order, so it is possible to speed up the nroff command by putting the most used characters first. For example:
The best way to create a terminal table for a new device is to take an existing terminal table and edit it to suit your needs. Once you create such a file, put it in the `/usr/share/lib/nterm` directory. Then, give it the name `tab.xyz`, where the `xyz` variable is the name of the terminal and also the name that you pass the `nroff` command by way of the `-T` flag. For example:

```
nroff -Txyz
```

**Files**

`/usr/share/lib/nterm/tab.Name` Contains terminal files.

**Related Information**

The `iconv` command, `nroff` command.

[International character support in text formatting] in *Operating system and device management* discusses the European-language extended character set and the commands that use it.

**Permissions File Format for BNU**

**Purpose**

Specifies BNU permissions for remote systems that call or are called by the local system.

**Description**

The `/etc/uucp/Permissions` file specifies access for remote systems that use the Basic Networking Utilities (BNU) program to communicate with the local system. The Permissions file contains an entry for each system the local system contacts using BNU. These entries correspond to entries in the `/etc/uucp/Systems` file or other systems files listed in the `/etc/uucp/Sysfiles` file with the same format. The Permissions file also contains an entry for each login ID that remote systems are permitted to use when using BNU to log into the local system.

Entries in the Permissions file specify:

- The login ID for a remote system
- The circumstances under which a remote system is allowed to send files to and receive files from the local system
- The commands a remote system is permitted to execute on the local system

The access permissions set in a Permissions file affect remote systems as a whole. They do **not** pertain to individual users who work on those remote systems. Permissions limiting `uucico` and `uuqt` daemon activities restrict the BNU access to a local system by **all** users on a specified remote system. The default permissions for sending and receiving files and executing commands are very restrictive. However, the file also provides options that enable you to change these defaults if you want to allow remote systems to have less restricted access to the local system.

Each entry in a Permissions file is a logical line. If an entry is too long to fit on the screen, make the last character in that physical line a `\` (backslash), which indicates continuation, and then type the remainder of the entry on the next physical line.
Each logical line contains a required entry specifying a login ID (LOGNAME entry) or the name of a remote system (MACHINE entry), followed by optional option/value pairs separated by either spaces or tabs. Both the LOGNAME and MACHINE entries and the option/value pairs are composed of name/value pairs. Name/value pairs consist of the name of the entry or option followed by an = (equal sign) and the value of the entry or option, with no spaces allowed within the pair.

The Permissions file can also contain comment lines and blank lines. Comment lines begin with a # (pound sign) and occupy the entire physical line. Blank lines are ignored.

Notes:
1. Access permissions set in the Permissions file affect all BNU communications, including those made through the mail facility or over a TCP/IP connection. Entries in a Permissions file do not affect a remote-system user with a valid login on a specified local system. Remote login commands (such as cu, ct, tn, or tip) connect to and log in on a system regardless of the restrictions set up in the local Permissions file. A user with a valid login ID is subject only to the permission codes established for that user's user ID (UID) and group ID (GID).
2. Examples of using the Permissions file are provided. The examples include issuing default or restricted access to remote systems and combining LOGNAME and MACHINE entries.

LOGNAME and MACHINE Entries

The Permissions file contains two types of required entries:

- **LOGNAME**
  Specifies the login IDs and access permissions for remote systems that are allowed to contact the local system.

- **MACHINE**
  Specifies the names and access permissions for the remote systems that the local system can contact.

Both LOGNAME and MACHINE entries specify what the remote system can do on the local system. LOGNAME entries take effect when a remote system contacts the local system. MACHINE entries take effect when the local system contacts a remote system. The permissions given to the remote system in the two types of entries can be the same or different.

For example, if remote system hera contacts local system zeus and logs in as uhera, the LOGNAME=uhera entry in the Permissions file on zeus controls what actions system hera can take on system zeus. If system zeus contacts system hera, the MACHINE=hera entry in the Permissions file on zeus controls what actions system hera can take on system zeus.

The most restrictive LOGNAME and MACHINE entry is an entry without any option/value pairs, which means that the remote system's access to the local system is defined by the default permissions. To override these defaults, include option/value pairs in the entry. The available options are:

- REQUEST
- SENDFILES
- READ, WRITE
- NOREAD, NOWRITE
- COMMANDS
- VALIDATE
- CALLBACK

These options allow different remote systems different types of access to the local system when using the BNU file transport and command execution programs. A LOGNAME and a MACHINE entry can be combined into a single entry when both include the same options.
LOGNAME Entry
A LOGNAME entry specifies one or more login IDs for remote systems permitted to log into the local system to conduct uucico and uuxqt daemon transactions, plus the access permissions for those remote systems. The login ID can be any valid login name. The LOGNAME entry specifies permissions for the remote system when it contacts the local system. The format of a LOGNAME entry is:

LOGNAME=LoginID[:LoginID . . .] [Option=Value . . .]

Remote systems log in with one of the IDs listed in the LoginID list. While logged in with that ID, the remote system has the permissions specified in the Option=Value list. The remote system that is calling must be listed in the /etc/uucp/Systems file or an alternative uucico service systems file specified in /etc/uucp/Sysfiles on the local system.

To specify more than one login ID with the same option/value pairs, list them in the same LOGNAME entry, separated by colons but without spaces. To specify multiple login IDs with different option/value pairs, list them in separate LOGNAME entries.

The most restrictive LOGNAME entry is an entry without any option/value pairs. The remote system’s access to the local system is then defined by these default permissions:

- The remote system cannot ask to receive any queued files from the local system.
- The local system cannot send queued work to the calling remote system when the remote system has completed its current operations. Instead, the queued work can be sent only when the local system contacts the remote system.
- The remote system cannot send files to (write) or transfer files from (read) any location except the BNU public directory (var/spool/uucppublic/SystemName) on the local system.
- Users on the remote system can execute only the default commands on the local system. (The default command set includes only the rmail command, which users implicitly execute by issuing the mail command.)

To override these defaults, include option/value pairs in the LOGNAME entry.

Note: A login ID can appear in only one LOGNAME entry. If there is a single entry for a login ID, that entry alone is sufficient for all remote systems using that login ID.

Attention: Allowing remote systems to log in to the local system with the uucp login ID seriously jeopardizes the security of your system. Remote systems logged in with the uucp ID can display and possibly modify (depending on the other permissions specified in the LOGNAME entry) the local Systems and Permissions files. It is strongly recommended that you create other BNU login IDs for remote systems and reserve the uucp login ID for the person responsible for administering BNU on the local system. Each remote system that contacts the local system should have a unique login ID with a unique UID.

MACHINE Entry
The Permissions file contains a MACHINE entry for each remote system the local system is permitted to contact. The access permissions specified in the MACHINE entry affect the remote system’s access to the local system when the local system contacts the remote system. Following is the format of a MACHINE entry:

MACHINE=SystemName[:SystemName . . .] [Option=Value . . .]

OR

MACHINE=OTHER [Option=Value . . .]

The most restrictive type of MACHINE entry, which uses the default permissions, is:
MACHINE=SystemName:[;SystemName . . ]

The system names are separated by a colon. The entry includes no spaces or tab characters. There are no option/value pairs, indicating that remote system access to the local system is defined by the following default permissions:

- The remote system cannot ask to receive any local system files queued to run on the calling remote system.
- The remote system cannot access (read) any files except those in the public directory on the local system.
- The remote system can send (write) files only to the local public directory.
- The remote system can execute only those commands in the default command set on the local system.

To override these defaults, include option/value pairs in the LOGNAME entry.

The SystemName list in a MACHINE entry may include a number of different remote systems. A MACHINE entry can also be:

MACHINE=OTHER [Option=Value . . ]

where the word OTHER represents a system name. This sets up access permissions for remote systems not specified in the existing MACHINE entries in a Permissions file. The MACHINE=OTHER entry is useful in these circumstances:

- When your installation includes a large number of remote systems that the local system regularly contacts for uucico and uuxqt daemon transactions
- When it is occasionally necessary to change the default command set specified in the COMMANDS option in the MACHINE entry.

Rather than create separate MACHINE entries for each of a large group of remote systems, set up one MACHINE=OTHER entry that includes the appropriate commands specified in a COMMANDS option entry. Then, when it becomes necessary to change the default command set, change the list of commands in only one entry rather than in numerous entries. Usually, a MACHINE=OTHER entry also specifies more restrictive option values for the unidentified remote systems.

Note: The local system cannot call any remote system that is not listed by name in a MACHINE entry, unless there is a MACHINE=OTHER entry in the Permissions file on the local system.

Option/Value Pairs

Option/value pairs can be used with the LOGNAME and MACHINE entries. The default permissions are restrictive, but can be changed with one or more of the option/value pairs. These options allow different remote systems different types of access to the local system when using the BNU file transport and command execution programs.

CALLBACK Option

The CALLBACK option, included in LOGNAME entries, specifies that no file transfer transactions will occur until the local system contacts the targeted remote system. The format of the CALLBACK option is either:

CALLBACK=no

OR

CALLBACK=yes

Note: If two systems both include the CALLBACK=yes option in their respective Permissions files, they cannot communicate with each other using BNU.
The default value, \texttt{CALLBACK=no}, specifies that the remote system may contact the local system and begin transferring files without the local system initiating the operations.

For tighter security, use the \texttt{CALLBACK=yes} option to specify that the local system must contact the remote system before the remote system may transfer any files to the local system.

If you include the \texttt{CALLBACK=yes} option in the \texttt{LOGNAME} entry, you must also have a \texttt{MACHINE} entry for that system so that your system can call it back. You can have a \texttt{MACHINE=OTHER} entry to allow your system to call any remote system, including the one for which the \texttt{CALLBACK=yes} option is specified.

The default value, \texttt{CALLBACK=no}, is generally sufficient for most sites.

**COMMANDS Option**

The \texttt{COMMANDS} option, included only in a \texttt{MACHINE} entry, specifies the commands that the remote systems listed in that \texttt{MACHINE} entry can execute on the local system. The format of the \texttt{COMMANDS} option is either:

\texttt{COMMANDS=CommandName[\texttt{::CommandName}} \ldots ]

OR

\texttt{COMMANDS=ALL}

The default is \texttt{COMMANDS=rmail:uucp}. Under the default, remote systems can run only the \texttt{rmail} and \texttt{uucp} commands on the local system. (Users enter the \texttt{mail} command, which then calls the \texttt{rmail} command.)

The commands listed in the \texttt{COMMANDS} option override the default. You can also specify path names to those locations on the local system where commands issued by users on remote systems are stored. Specifying path names is useful when the default path of the \texttt{uuxqt} daemon does not include the directory where a command resides.

\textbf{Note:} The default path of the \texttt{uuxqt} daemon includes only the \texttt{/usr/bin} directory.

To allow a certain remote system to execute all available commands on the local system, use the \texttt{COMMANDS=ALL} format. This specifies that the command set available to the designated remote system includes all commands available to users on the local system.

\textbf{Note:} The \texttt{COMMANDS} option can jeopardize the security of your system. Use it with extreme care.

**NOREAD and NOWRITE Options**

The \texttt{NOREAD} and \texttt{NOWRITE} options, used in both \texttt{LOGNAME} and \texttt{MACHINE} entries, delineate exceptions to the \texttt{READ} and \texttt{WRITE} options by explicitly forbidding access by the remote system to directories and files on the local system.

The formats of these options follow:

\texttt{NOREAD=PathName[::PathName} \ldots ]

\texttt{NOWRITE=PathName[::PathName} \ldots ]

\textbf{Note:} The specifications you enter with the \texttt{READ}, \texttt{WRITE}, \texttt{NOREAD}, and \texttt{NOWRITE} options affect the security of your local system in terms of BNU transactions.
READ and WRITE Options
The READ and WRITE options, used in both LOGNAME and MACHINE entries, specify the path names of directories that the uucico daemon can access when transferring files to or from the local system. You can specify more than one path for uucico daemon activities.

The default location for both the READ and WRITE options is the /var/spool/uucppublic directory (the BNU public directory) on the local system. The formats for these options follow:

`READ=PathName[ PathName ...]`

`WRITE=PathName[ PathName ...]`

The source file, destination file, or directory must be readable or writable for the other group for the BNU program to access it. Set these permissions with the chmod command. A user without root user authority can take away permissions granted by the READ and WRITE options, but that user cannot grant permissions that are denied by these options.

If the READ and WRITE options are not present in the Permissions file, the BNU program transfers files only to the /var/spool/uucppublic directory. However, if you specify path names in these options, enter the path name for every source and destination, including the /var/spool/uucppublic directory if the remote system is to be permitted access to it.

Attention: Specifications with the READ, WRITE, NOREAD, and NOWRITE options affect the security of your local system in terms of BNU transactions. The subdirectories of directories specified in the READ and WRITE options can also be accessed by the remote system unless these subdirectories are forbidden with the NOREAD or NOWRITE options.

REQUEST Option
The REQUEST option, used in both LOGNAME and MACHINE entries, enables a remote system to ask to receive any queued files containing work that users on the local system have requested to be executed on that remote system. The default is not to allow such requests.

When a remote system contacts the local system to transfer files or execute commands, the remote system may also request permission to receive any files queued on the local system for transfer to or execution on that remote system. This format of the REQUEST option permits such requests:

`REQUEST=yes`

The default, REQUEST=no, does not have to be entered. This specifies that the remote system cannot ask to receive any work queued for it on the local system. The local system must contact the remote system before transmitting files and execute commands queued on the local system to the remote system.

Use the REQUEST=yes option in both LOGNAME and MACHINE entries to allow remote-system users to transfer files to and execute commands on a local system on demand. Restrict access with the REQUEST=no option so that the local system retains control of file transfers and command executions initiated by remote systems.

Note: Entries in the Permissions file affect only BNU transactions. They do not affect remote-system users with valid logins on a local system.

SENDFILES Option
The default allows the local system to transfer queued work to the remote system only when the local system contacts the remote system. However, when a remote system finishes transferring files to or executing commands on a local system, that local system may try to send queued work to the calling remote system immediately. To enable an immediate transfer, use the following SENDFILES option:
SENDFILES=yes

The SENDFILES=yes option allows the transfer of queued work from the local to the remote system once the remote system has completed its operations. The default value, SENDFILES=call, specifies that local files queued to run on the remote system are sent only when the local system contacts the remote system.

Notes:
1. The SENDFILES option is ignored when it is included in a MACHINE entry.
2. Entries in the Permissions file affect only BNU transactions. They do not affect remote-system users with valid logins on a local system.

VALIDATE Option

The VALIDATE option provides more security when including commands in the default command set that could cause damage when executed by a remote system on a local system. Use this option, specified only in a MACHINE entry, in conjunction with a COMMANDS option. The format of the VALIDATE option is:

 VALIDATE=LoginName[: LoginName . . .]

The VALIDATE option verifies the identity of the calling remote system. Including this option in a MACHINE entry means that the calling remote system must have a unique login ID and password for file transfers and command executions.

Note: This option is meaningful only when the login ID and password are protected. Giving a remote system a special login and password that provide unlimited file access and remote command-execution ability is equivalent to giving any user on that remote system a normal login and password on the local system, unless the special login and password are well-protected.

The VALIDATE option links a MACHINE entry, which includes a specified COMMANDS option, to a LOGNAME entry associated with a privileged login. The uuxqt daemon, which executes commands on the local system on behalf of users on a remote system, is not running while the remote system is logged in. Therefore, the uuxqt daemon does not know which remote system sent the execution request.

Each remote system permitted to log in to a local system has its own spooling directory on that local system. Only the BNU file transport and command execution programs are allowed to write to these directories. For example, when the uucico daemon transfers execution files from the remote system hera to the local system zeus, it places these files in the /var/spool/uucppublic/hera directory on system zeus.

When the uuxqt daemon attempts to execute the specified commands, it determines the name of the calling remote system (hera) from the path name of the remote-system spooling directory (/var/spool/uucppublic/hera). The daemon then checks for that name in a MACHINE entry in the Permissions file. The daemon also checks for the commands specified in the COMMANDS option in a MACHINE entry to determine whether the requested command can be executed on the local system.

Security

Access Control: Only a user with root authority can edit the Permissions file.

Examples

The following are examples of using the Permissions file.

Providing Default Access to Remote Systems

1. To provide the default permissions to any system logging in as uucp1, enter:

   LOGNAME=uucp1

2. To provide the default permissions to systems venus, apollo, and athena when called by the local system, enter:
Providing Less Restricted Access to Remote Systems

1. The following LOGNAME entry allows remote system merlin to read and write to more directories than just the spool directory:

   LOGNAME=umerlin READ=/ NOREAD=/etc:/usr/sbin/uucp
   WRITE=/home/merlin:/var/spool/uucppublic

   A system logging in as user umerlin can read all directories except the /usr/sbin/uucp and /etc directories, but can write only to the /home/merlin and public directories. Because the login name umerlin has access to more information than is standard, BNU validates the system before allowing merlin to log in.

2. The following example allows remote system hera unrestricted access to system zeus, and shows the relationship between the LOGNAME and MACHINE entries:

   LOGNAME=uhera REQUEST=yes SENDFILES=yes
   READ=/ WRITE=/MACHINE=hera VALIDATE=uhera REQUEST=yes \COMMANDS=ALL READ=/ WRITE=/

The remote system hera may engage in the following uucico and uuxqt transactions with system zeus:

- System hera may request that files be sent from system zeus, regardless of which system placed the call (REQUEST=yes appears in both entries);
- System zeus may send files to system hera when system hera contacts system zeus (SENDFILES=yes in the LOGNAME entry);
- System hera may execute all available commands on system zeus (COMMANDS=ALL in the MACHINE entry);
- System hera may read from and write to all directories and files under the root directory on system zeus, regardless of which system placed the call (READ=/ WRITE=/ in both entries).

Because the entries provide system hera with relatively unrestricted access to system zeus, BNU validates the log name before permitting system hera to log in.

Note: This entry allows unrestricted access to the local system by the remote system listed in the MACHINE entry. This entry can jeopardize the security of your system.

Combining LOGNAME and MACHINE Entries

1. Following are LOGNAME and MACHINE entries for system hera:

   LOGNAME=uhera REQUEST=yes SENDFILES=yes
   MACHINE=hera VALIDATE=uhera REQUEST=yes COMMANDS=rmail:news:uucp

   Since they have the same permissions and apply to the same remote system, these entries can be combined as:

   LOGNAME=uhera SENDFILES=yes REQUEST=yes \ MACHINE=hera VALIDATE=uhera COMMANDS=rmail:news:uucp

2. LOGNAME and MACHINE entries used for more than one remote system can be combined if they have the same permissions. For example:

   LOGNAME=uucp1 REQUEST=yes SENDFILES=yes
   MACHINE=zeus:apollo:merlin REQUEST=yes COMMANDS=rmail:uucp

   can be combined as:

   LOGNAME=uucp1 REQUEST=yes SENDFILES=yes \ MACHINE=zeus:apollo:merlin COMMANDS=rmail:uucp

Either form of the entries allows systems zeus, apollo, and merlin the same permissions. They can:

- Log into the local system as uucp1.
- Execute the rmail and uucp commands.
- Request files from the local system, regardless of which system placed the call.
Allowing Access to Unnamed Systems
To allow your system to call systems that are not specified by name in a MACHINE entry, use a
MACHINE=OTHER entry as follows:
MACHINE=OTHER COMMANDS=rmail

This entry allows your system to call any machine. The machine called will be able to request execution of
the rmail command. Otherwise, the default permissions apply.

Permissions File Entries for Three Systems
The following examples show the Permissions files for three connected systems:

On system venus:
LOGNAME=uhera MACHINE=hera \nREAD=/ WRITE=/ COMMANDS=ALL \nNOREAD=/usr/secure:/etc/uucp \nNOWRITE=/usr/secure:/etc/uucp \nSENDFILES=yes REQUEST=yes VALIDATE=hera

On system hera:
LOGNAME=uvenus MACHINE=venus \nREAD=/ WRITE=/ COMMANDS=rmail:who:lp:uucp \nSENDFILES=yes REQUEST=yes

LOGNAME=uucp1 MACHINE=OTHER \nREQUEST=yes SENDFILES=yes

On system apollo:
LOGNAME=uhera MACHINE=hera \nREAD=/var/spool/uucppublic:/home/hera \nREQUEST=no SENDFILES=call

Given these permissions:
- System hera logs into system venus as uhera. It can request or send files regardless of who initiated the
call and can read or write to all directories except /usr/secure and /usr/sbin/uucp. It can execute any
command. However, before system venus allows any system to log in as uhera, it checks to make sure
that system is hera.
- System venus logs into system hera as uvenus. After it logs in, it can read or write to all directories on
system hera and can request or send commands regardless of who initiated the call. It can execute the
rmail, who, lp, and uucp commands only.
- System hera logs into system apollo as uhera. After it logs in, it can send files, but requests to receive
files will be denied. It can read and write only from the public directory and the /home/hera directory,
and can execute only the default list of commands.
- System apollo logs into system hera as uucp1, since it does not have a unique login ID on system
hera. It can request and send files, regardless of who initiated the call. It can read and write only from
the public directory (the default) and execute only the default list of commands.

Note: The uucp1 login ID defined on system hera can be used by any remote system, not just by
system apollo. In addition, the presence of the MACHINE=OTHER entry allows system hera to
call machines not specified elsewhere in the Permissions file. If system hera calls an unknown
machine, the permissions in the MACHINE=OTHER entry take effect.

Files
/etc/uucp/Permissions file Describes access permissions for remote systems.
/etc/uucp/Systems file Describes accessible remote systems.
/etc/uucp/Sysfiles file

Specifies possible alternative files for the /etc/uucp/Systems file.

/var/spool/uucppublic directory

Contains files that have been transferred.

Related Information

The chmod command, mail command, rmail command, uuchek command, uucpaadm command.

The uucico daemon and uuxqt daemon read the Permissions file.

Configuring BNU | Understanding the BNU File and Directory Structure | Understanding BNU Security in Networks and communication management.

phones File Format for tip

Purpose

Describes connections used by the tip command to contact remote systems.

Description

The /etc/phones-file file lists the remote systems that can be contacted using the tip command, along with the telephone numbers used to contact those systems.

A sample phones-file file for the tip command is included with the operating system. The sample file is named /usr/lib/phones-file. A user with root user authority can copy the sample file to the /etc/phones file and modify it to suit the needs of a particular site.

Any tip user can create an individual phones file in the format of the phones-file file. The individual phones file can be named with any operating system file name and placed in any directory to which the user has access. To instruct the tip command to use the new file, either set the tip command phones variable or set an environment variable named PHONES.

Systems listed in the phones file must also be described in the /etc/remote-file file, in the file specified by the REMOTE environment variable, or in the file specified by the tip command remote variable.

Format of Entries

The format of an entry in the phones file is:

SystemName    PhoneNumber

The SystemName field and the PhoneNumber field must be separated by at least one space. More than one space can be used to improve readability.

SystemName    Specifies the name of the remote system to be contacted.
PhoneNumber    Specifies the telephone number, including line access codes, to be used to reach the remote system. Dashes may be used for readability.

If more than one phone number can be used to reach a certain system, make multiple entries for that system, placing each entry on a separate line.

Any line beginning with a # (pound sign) is interpreted as a comment.

Examples

1. To list phone numbers in a phones file, make entries similar to the following:
System hera is contacted using the telephone number 123-7654. To contact system zeus, a line-access code of 9 is followed by the telephone number 512-345-9999.

2. To define more than one phone number for the same system, make multiple entries for that system, as follows:

decvax 9-915-987-1111
decvax 9-915-987-2222

If the tip command cannot reach the decvax system using the first phone number, it attempts to contact the system using the second phone number.

Files

/etc/phones Denotes complete path name of the phones file.
/usr/lib/phones-file Contains an example phones file.
/etc/remote Describes remote systems that can be contacted using the tip command.

Related Information

The [tip] command.

The Communication with connected UNIX systems using the tip command in Networks and communication management.

Poll File Format for BNU

Purpose

Specifies when the BNU program should poll remote systems.

Description

The /etc/uucp/Poll file specifies when the Basic Networking Utilities (BNU) program should poll (initiate automatic calls to) designated remote computers. This file is used in conjunction with the /var/spool/cron/crontabs/uucp file, uudemon.hour command, and uudemon.poll command. Together, these files are responsible for initiating automatic calls to certain remote systems.

Each entry in the Poll file contains the name of the remote computer followed by a sequence of times when the BNU program should poll that system. Modify the times specified in the Poll file based on how the systems at your site are used. Specify times as digits between 0 and 23. The format of the entry is as follows:

SystemName Time [Time ...]

The fields in the Poll file entry must be separated by at least one space. More spaces can be used for readability. A tab character between the SystemName field and the first Time field is optional.

Notes:

1. Only someone with root user authority can edit the Poll file, which is owned by the uucp program login ID.

2. Most versions of UUCP require a tab character between the SystemName field and the first Time field. In BNU, either a tab or spaces will work.
Examples
Following is a standard entry in the Poll file:
hera <TAB> 0 4 8 12 16 20

This entry specifies that the local system will poll the remote system hera every 4 hours.

The tab character can be replaced by one or more spaces. Thus the preceding entry is equivalent to the following one:
hera 0 4 8 12 16 20

Files
/etc/locks
Contains lock files that prevent multiple uses of devices and multiple calls to systems.
/var/spool/cron/crontabs/uucp
Schedules BNU jobs for the cron daemon.

Related Information
The uucpadm command, uudemon.hour command, uudemon.poll command.
The cron daemon.
Configuring BNU Setting Up BNU Polling of Remote Systems Understanding the BNU File and Directory Structure in Networks and communication management.

profile File Format

Purpose
Sets the user environment at login time.

Description
The $HOME/.profile file contains commands that the system executes when you log in. The .profile also provides variable profile assignments that the system sets and exports into the environment. The /etc/profile file contains commands run by all users at login.

After the login program adds the LOGNAME (login name) and HOME (login directory) variables to the environment, the commands in the $HOME/.profile file are executed, if the file is present. The .profile file contains the individual user profile that overrides the variables set in the profile file and customizes the user-environment profile variables set in the /etc/profile file. The .profile file is often used to set exported environment variables and terminal modes. The person who customizes the system can use the mkuser command to set default .profile files in each user home directory. Users can tailor their environment as desired by modifying their .profile file.

Note: The $HOME/.profile file is used to set environments for the Bourne and Korn shells. An equivalent environment for the C shell is the $HOME/.cshrc file.

Examples
The following example is typical of an /etc/profile file:

#Set file creation mask unmask 022
#Tell me when new mail arrives
MAIL=/usr/mail/$LOGNAME
#Add my /bin directory to the shell
#search sequence

546 AIX 5L Version 5.3 Files Reference
#Set terminal type
TERM=lft
#Make some environment variables global
export MAIL PATH TERM

## Files

/\etc/\profile Contains profile variables.

## Related Information

The `bsh` command, `csh` command, `env` command, `login` command, `mail` command, `mkuser` command, `ksh` command, `stty` command, `su` command.

The Profiles overview in *Operating system and device management* discusses profiles and how they can be modified for individual needs.

The Shells in *Operating system and device management* describes what shells are, the different types, and how they affect the way commands are interpreted.

### protocols File Format for TCP/IP

#### Purpose

Defines the Internet protocols used on the local host.

#### Description

The `/\etc/\protocols` file contains information about the known protocols used in the DARPA Internet. Each protocol is represented by a single line in the `protocols` file. Each entry corresponds to the form:

Name  Number  Aliases

The fields contain the following information:

- **Name**: Specifies an official Internet Protocol name.
- **Number**: Specifies a protocol number.
- **Aliases**: Specifies any unofficial names used for the protocol.

Items on a line are separated by one or more spaces or tab characters. Comments begin with the # (pound sign), and routines that search the `protocols` file do not interpret characters from the beginning of a comment to the end of the line. A protocol name can contain any printable character except a field delimiter, new line character, or comment character.

The lines appear as follows:

```
ip      0  #dummy for IP
icmp    1  #control message protocol
#gpp    2  #gateway^2 (not normally used)
tcp     6  #tcp
tgpp    8  #exterior gateway protocol
tpup    12 #tpup
udp     17 #user datagram protocol
```

---

**Chapter 2. File Formats** 547
Related Information

The `getprotoent` subroutine. TCP/IP protocols in *Networks and communication management.*

queuedefs File Format

**Purpose**
Specifies the handling of `cron` daemon events.

**Description**
The `/var/adm/cron/queuedefs` file defines how the system handles different `cron` daemon events types. The file specifies the maximum number of processes per event type to schedule at one time, the nice value of the event type, and how long to wait before retrying to execute a process. The following event types can be scheduled by the `cron` daemon:

- `at` command events
- `batch` command events
- `crontab` command events
- `sync` subroutine events
- `ksh` command events
- `csh` command events

This file is empty as shipped, but can be modified to change how the `cron` daemon handles each event type. Each entry in the `queuedefs` file is of the form:

```
EventType.[Jobsj][Nicen][Waitw]
```

The fields are described as follows:

- **EventType**: Specifies a character representing an event type. The following are valid values for the `EventType` field:
  - `a`: Specifies an `at` command event.
  - `b`: Specifies a `batch` command event.
  - `c`: Specifies a `crontab` command event.
  - `d`: Specifies a `sync` subroutine event.
  - `e`: Specifies a `ksh` command event.
  - `f`: Specifies a `csh` command event.

- **Jobsj**: Specifies the maximum number of jobs the `cron` daemon can start at one time. The default value is 100.

- **Nicen**: Specifies the nice value for job execution. The default value is 2.

- **Waitw**: Specifies the time, in seconds, to wait before attempting to execute the command again. The default value is 60 seconds.

**Note:** You must have root user authority to modify this file.

The `at` command allows you to specify the time when a command should be run. Each command or program will be assigned a job number and will be queued in the `/var/spool/cron/atjobs` directory.

The queueing system may also be set up by defining a batch queue in the `/etc/qconfig` file and using the `enq` command to submit a job to this queue. This queue may be set up with a first-come, first-serve discipline. The following stanzas should be added to the `/etc/qconfig` file to enable this:
bsh
device = bshdev
discipline = fcfs
bshdev:
backend = usr/bin/sh

This configuration may already exist in the /etc/qconfig file. If you want your commands and programs to run under the Korn shell, you should change the last line in the above stanza to:
backend = usr/bin/ksh

After creating the above stanza in the /etc/qconfig file, enable the queue by issuing the following:
qchk -A

Programs and commands may now be run on a first-come, first-serve basis using the enq command. For example, to run the program PROGRAM1 from the bsh queue, enter:
enq -P bsh PROGRAM1

The flags for the batch facility and queueing are:

- **at -qa**: This is for queueing at jobs.
- **at -qb**: This is for queueing batch jobs.
- **at -qe**: This is for queueing ksh jobs.
- **at -qf**: This is for queueing csh jobs.

**Examples**
1. To set the at command job queue to handle 4 concurrent jobs with a nice value of 1 and no retries, enter:
a.4j1n
2. To set the crontab command job queue to handle 2 concurrent jobs with a nice value of 2 and a retry in 90 seconds if the fork subroutine fails, enter:
c.2j2n90w

**Related Information**
The `at` command, `batch` command, `crontab` command, `csh` command, `enq` command, `ksh` command, `rc` command.

The `cron` daemon.

The `fork` subroutine, `sync` subroutine.

**rc.net File Format for TCP/IP**

**Purpose**
Defines host configuration for network interfaces, host name, default gateway, and static routes.

**Description**
The `/etc/rc.net` file is a shell script that contains configuration information. The stanzas allow you to enable the network interfaces and set the host name, the default gateway, and any static routes for the current host. This file can be used as a one-step configuration alternative to using individually the set of commands and files necessary to configure a host.
The rc.net shell script is run by the configuration manager program during the second phase of configuration. If TCP/IP is installed, a second script, rc.tcpip, is run from the init command after the second phase of configuration has completed and after the init command has started the SRC master.

Stanzas in the file should appear in the order in which they are presented here.

The rc.net shell script may also be run by the configuration manager program (cfgmgr) if cfgmgr is run after system configuration is completed. It is often run at other times to configure new devices that have been added to the system since boot time. If cfgmgr runs rc.net, both the configuration methods and rc.net itself check to see if networking devices are already in the Available state. If so, the values of device attributes are not changed to avoid overwriting any configuration changes that have been made since boot time.

If /etc/rc.net is run without cfgmgr, device attributes will be reset to the values in the ODM database regardless of the states of the devices. This allows a system’s configuration to be restored to the values specified in the ODM database.

**Using the Configuration Methods**

These stanzas use the configuration methods for TCP/IP to manipulate the ODM database.

**Configuring Network Interfaces**

For each network adapter that has been previously configured, a set of stanzas is required. The following stanzas define, load, and configure the appropriate network interfaces for every configured network adapter. These configuration methods require that the interface and protocol information be entered in the ODM database, using either SMIT or high-level configuration commands such as the mkdev command. The network interface configuration information is held in the running system only and must be reset at each system restart.

```bash
/usr/lib/methods/defif >> $LOGFILE 2>&1
/usr/lib/methods/cfgif $* >> $LOGFILE 2>&1
```

The defif method defines the network interfaces. The cfgif method configures the network interfaces in the configuration database.

The second part of the stanzas indicates that output should be sent to a log file. The log file must include the full path name. If no log file is specified, the default log file is /dev/null.

Along with the network interface configuration, additional commands must be executed for X.25 and SLIP interfaces: the x25ip command for X.25 interfaces and the slattach command for SLIP connections. The x25ip command loads the X.25 translation table into the kernel and the slattach command is used to assign a TTY line to an interface for SLIP. For each SLIP interface, the slattach command must be executed for the appropriate TTY.

At times, when diskless clients reboot using these configuration methods they hang on LED 581. This happens because diskless clients use server disk space to store the logging information. To get the client to reboot when this happens, execute the /usr/lib/methods/cfgif configuration method in the client rc.net file that resides on the server without message logging as follows:

```
/usr/lib/methods/cfgif $*
```

**Setting the Host Name, Default Gateway, and Any Static Routes**

The following stanzas set the host name, default gateway, and static routes, using the definet and cfginet subroutines to alter the ODM database for the inet0 object.

```
/usr/lib/methods/definet >> $LOGFILE 2>&1 /usr/lib/methods/cfginet >> $LOGFILE 2>&1
```

550   AIX 5L Version 5.3 Files Reference
The second part of the stanzas indicates that output should be sent to a log file. The log file must include the full path name. If no log file is specified, the default log file is /dev/null.

**Using Traditional Configuration Commands**

These stanzas use configuration commands for TCP/IP to set configuration values.

### Configuring Network Interfaces

The following stanza defines, loads, and configures the specified network interface:

```
/usr/sbin/ifconfig Interface inet
InternetAddress up>>$LOGFILE 2 &1
```

The *Interface* parameter should specify the type and number of the interface, for example, tr0. The *InternetAddress* parameter should specify the Internet address of the interface, for example, 192.1.8.0.

The last part of the stanza indicates that output should be sent to a log file. The log file must include the full path name. If no log file is specified, the default log file is /dev/null.

### Setting the Host Name, Default Gateway, and Any Static Routes

These stanzas should follow any stanzas for the network interfaces. These stanzas use the hostname command to set the host name and the route command to define the default gateway and any static routes. The static route information is held in the running system only and must be reset at each system restart.

```
/usr/bin/hostname  Hostname >>
$LOGFILE 2>&1/usr/sbin/route add 0
Gateway >> $LOGFILE 2>&1
/usr/sbin/route add DestinationAddress
Gateway >>$LOGFILE 2>&1
```

The *add* variable for the route command adds a static route to the host. This route can be to the default gateway (by specifying a hop count, or metric, of 0), or to another host through a gateway.

The last part of the stanzas indicates that output should be sent to a log file. The log file must include the full path name. If no log file is specified, the default log file is /dev/null.

### Miscellaneous Functions

Use these stanzas to set the host ID and user name. By default, the host ID and user name are set to the host name. However, these stanzas can be altered to customize the host ID and user name.

```
/usr/sbin/hostid `hostname`
/usr/bin/uname -s `hostname | sed -e 's/\..*$//'` 
>> $LOGFILE 2>&1
```

To customize these stanzas, replace the hostname entry in single quotation marks with the desired host ID or user name.

The second part of the user name stanza indicates that output should be sent to a log file. The log file must include the full path name. If no log file is specified, the default log file is /dev/null.

### Load Network File System (NFS)

If you have the Network File System (NFS) installed on the current host, the following stanza loads and configures the NFS kernel extension:

```
if [-x /usr/sbin/gfsinstall -a 
-x /usr/lib/drivers/nfs.ext ] ; then 
    /usr/sbin/gfsinstall -a /usr/lib/drivers/
    nfs.ext >>$LOGFILE 2>&1fi
```
The last part of the NFS stanza indicates that output should be sent to a log file. The log file must include the full path name. If no log file is specified, the default log file is /dev/null.

Examples
1. To set up a Token-Ring interface, using the `ifconfig` command, include the following stanza:

   ```
   /usr/sbin/ifconfig tr0 inet
   192.1.8.0 up >>$LOGFILE 2>&1
   ```

   This stanza defines Token-Ring interface tr0, with the Internet address 192.1.8.0.

2. To set the host name, using the `hostname` command, include the following stanza:

   ```
   /usr/bin/hostname robo.austin.century.com
   >>$LOGFILE 2>&1
   ```

   This stanza sets host name robo.austin.century.com. The host name in this example includes domain and subdomain information, which is necessary if the host is using the domain naming system.

3. To set up a default gateway, using the `route` command, include the following stanza:

   ```
   /usr/sbin/route add 0
   192.100.13.7 >>$LOGFILE 2>&1
   ```

   The value 0 for the Metric parameter means that any packets sent to destinations not previously defined and not on a directly connected network go through the default gateway. The 192.100.13.7 address is the default gateway.

4. To set up a static route, using the `route` command, include the following stanza:

   ```
   /usr/sbin/route add net
   192.100.201.7 192.100.13.7 >>$LOGFILE 2>&1
   ```

   The 192.100.201.7 address is the receiving computer (the Destination parameter). The 192.100.13.7 address is the routing computer (the Gateway parameter).

Files

```
/etc/rc.tcpip
```

Initializes daemons at each system restart.

Related Information

The `hostname` command, `ifconfig` command, `init` command, `mkdev` command, `route` command, `sendmail` command, `slattach` command.

The `cfgif` method, `cfginet` method, `defif` method, `definet` method.

The `rc.tcpip` file.

The `inetd` daemon.

Installation of TCP/IP in Networks and communication management.

rc.ntx File Format

Purpose

Supplies configuration information for the Network Terminal Accelerator adapter card.
Description

The /etc/rc.ntx file invokes the hty_load command to load the /etc/hty_config file. This file can also specify a route to a gateway, using the ntx_route command. Also, the rc.ntx file enables SNMP.

The /etc/rc.ntx file can be used to perform different configuration tasks. For example, to supply a route to an additional gateway, add the following line immediately after the comment about additional routes, and supply an IP address for the Destination and Gateway parameters:

```
/usr/bin/ntx_route -drhp$i net Destination Gateway
```

Following is the file as it is shipped with the software package. You can add additional commands to the file, as indicated above.

```
echo "Executing hty_load"
/usr/bin/hty_load -f /etc/hty_config
echo "Finished executing hty_load"
```

```#!/ Maximum number of Network Terminal Accelerator adapters
# supported on each workstation.
#
MAX_RHP_DEVICES=7

i=0
while [ $i -le $MAX_RHP_DEVICES ]
doi [ -f /etc/rhp$i.ntx_comun.conf ]; then
echo "Configuring SNMP communities on NTX Adapter rhp$i"
/usr/bin/ntx_comun -d /dev/rhp$i -f /etc/rhp$i.ntx_comun.conf
fi
if [ -f /etc/rhp$i.ntx_traps.conf ]; then
echo "Configuring SNMP traps on NTX Adapter rhp$i"
/usr/bin/ntx_traps -d /dev/rhp$i -f /etc/rhp$i.ntx_traps.conf
fi
if [ -f /etc/rhp$i.ntx_nms.conf ]; then
echo "Configuring SNMP nms on NTX Adapter rhp$i"
/usr/bin/ntx_nms -d /dev/rhp$i -f /etc/rhp$i.ntx_nms.conf
fi
if [ -f /etc/rhp$i.ntx_descr.conf ]; then
echo "Configuring SNMP site-specific variables on NTX Adapter rhp$i"
/usr/bin/ntx_descr -d /dev/rhp$i -f /etc/rhp$i.ntx_descr.conf
fi
if [ -c /dev/rhp$i ]; then
  STATE=`lsattr -E -l rhp$i -a snmp -F value`
echo "Turning $STATE SNMP on NTX Adapter rhp$i"
/usr/bin/ntx_snmp -d /dev/rhp$i $STATE
fi
# Additional routes for each NTX Adapter can be added here
# example: /usr/bin/ntx_route -d /dev/rhp$i X.X.X X.X.X.X
i=`expr $i + 1`
```

Related Information

The hty_load command.
remote File Format for tip

Purpose
Describes remote systems contacted by the tip command.

Description
The /etc/remote-file file describes the remote systems that can be contacted using the tip command. When a user invokes the tip command, the command reads the remote file to find out how to contact the specified remote system. If invoked with the SystemName parameter, the tip command searches the remote file for an entry beginning with that system name. If invoked with the PhoneNumber parameter, the command searches the remote file for an entry beginning with tipBaudRate, where BaudRate designates the baud rate to be used for the connection.

Any tip user can create an individual remote file in the format of the remote file. The individual remote file can be named with any operating system file name and placed in any directory to which the user has access. To instruct the tip command to use the new file, set the REMOTE environment variable before issuing the tip command, or use the tip command remote variable.

A sample remote file for tip is included with the operating system. The sample file is named /usr/lib/remote-file. This sample file contains two examples, either of which is a complete remote file. One of the examples uses a set of general dialer definitions, followed by general system definitions, and specific systems. The second example defines each system individually.

Any user can copy the sample file to some other directory and modify it for individual use. A user with root user authority can copy the sample file to the /etc/remote file and modify it to suit the needs of a particular site.

Format of Entries
The general format of an entry in the /etc/remote-file file is a system name, baud rate, or dialer name followed by a description and one or more attributes, as follows:

SystemName[|SystemName ...]| Description:Attribute[:Attribute ...]:

OR

tipBaudRate|Description: Attribute[:Attribute ...]:

OR

DialerName[|DialerName ...]| Description:Attribute[:Attribute ...]:

The name of the system or dialer is followed by a | (pipe symbol) and a description of the system or dialer. More than one system or dialer name can be given; in this case, they must be separated by pipe symbols and precede the Description parameter. The last section in this list is always treated by the tip command as a description, not a system name.

The Description field is followed by a : (colon) and a list of attributes separated by colons. Each entry must also end with a colon.

An entry can be continued on the next line by typing a \ (backslash). The continuation line must begin with a : (colon) and can be indented for readability.

Any line beginning with a # (pound sign) is read as a comment line.
Note: Spaces can be used only within the Description parameter or in comment lines.

Attributes Used to Define Systems and Dialers
Use the following attributes to describe systems in the remote file:

```
at=ACUType
```
Defines the type of automatic calling unit (also known as the ACU or modem). This attribute should be specified in each entry (or in another entry included with the tc attribute) unless the system is linked to a modem. The ACUType must be one of the following:

- biz31f
- biz31w
- bix22f
- bix22w
- df02
- df03
- dn11
- ventel
- hayes
- courier
- vadic
- v3451
- v831

```
br#BaudRate
```
Specifies the baud rate to be used on the connection. The default rate is 1200 baud. This attribute should be specified in each entry or in another entry included with the tc attribute. The baud rate specified can be overridden using the tip command \[BaudRate\] parameter.

```
cu=Device
```
Specifies the device for the call unit if it is different from the device defined in the dv statement. The default is the device defined in the dv statement.

```
du
```
Makes a call. This attribute must be specified in each entry or in another entry included with the tc attribute.

```
dv=Device[Device ...]
```
Lists one or more devices to be used to link to the remote system. If the first device listed is not available, the tip command attempts to use the next device in the list, continuing until it finds one available or until it has tried all listed devices.

This attribute must be specified in each entry or in another entry included with the tc attribute.

```
el=Mark
```
Defines the mark used to designate an end-of-line in a file transfer. This setting is the same as that defined by the tip command \[eol\] variable.

```
fs=Size
```
Specifies the frame size. The default is the value of the BUFSIZE environment variable. This value can also be changed using the tip command \[framesize\] variable.

```
ie=InputString
```
Specifies the input end-of-file mark. The default setting is null value.

```
oe=OutputString
```
Specifies the output end-of-file mark. The default setting is a null value.

```
pa=Parity
```
Specifies the required parity setting for connecting to the remote system. The default setting is Even. Valid choices are: Even (7 bits, even parity), Odd (7 bits, odd parity), None (7 bits, no parity), and Graphic (8 bits, no parity).
pn=

Lists telephone numbers to be used to call the remote system. This entry is required if a modem is used to call a remote system, except in a tipBaudRate entry when a telephone number is entered with the tip command.

If the tip command is invoked with the PhoneNumber parameter, the pn attribute in the appropriate tipBaudRate entry is ignored and the number given when the command is invoked is used instead.

The pn attribute can be in either of the following forms:

\[ \text{pn=}@ \quad \text{Instructs tip to search the /etc/phones-file file, or the file specified with the \text{phones} variable, for the telephone number.} \]

\[ \text{pn=}Number[,Number...] \quad \text{Lists one or more phone numbers to be used to call the remote system.} \]

tc=Entry

Refers to another entry in the file. This allows you to avoid defining the same attributes in more than one entry. If used, this attribute should be at the end of the entry.

tc=DialerName

Includes the specified DialerName entry. The DialerName entry must be defined elsewhere in the remote file.

tc=SystemName

Includes the specified SystemName entry. The SystemName entry must be defined elsewhere in the remote file.

Setting Up Group Entries

Set up entries in the remote file in two ways. Define each system individually, giving all of its attributes in that entry. This works well if you are contacting several dissimilar systems.

Or [group the systems by similarity] To do this, use two or three groups, depending on how the systems are similar. The groups can be arranged by:

- Dialer definitions, including the device, baud rate, call unit, ACU type, and dial-up flag.
- General system definitions, including any information that several systems have in common. Use the tc attribute to refer to a dialer entry.
- Specific system descriptions, which use the tc attribute to refer to one of the general system types or a dialer entry.

You can omit either the dialer definitions or the general system definitions, depending on the way the remote systems are grouped.

Examples

Defining a System Individually

To define a system without using the tc= attribute, enter:

\[ \text{vms750|ghost|NPG} \]

\[ :dv=/dev/tty36,/dev/tty37:br#9600:el=^Z^U^C^S^Q^O:\]

\[ :ie=$@:oe=^Z:\]  

This entry defines system vms750, which can also be referred to as ghost. The system can be accessed using either /dev/tty36 or /dev/tty37, at a baud rate of 9600. The end-of-line mark is \(^Z^U^C^S^Q^O^\). The input end-of-file mark is $@ and the output end-of-file mark is ^Z. Since no phone number is defined, the system is accessed over a direct connection.

Grouping Systems by Similarity

The following examples use a dialer entry and a general system entry, followed by specific system entries that refer to the general entries.

1. To define a dialer, enter:
This entry defines a dialer called dial1200. The dialer is connected to device /dev/cul1 and is an ACU type of dn11. The dial-up (du) flag is set.

2. To define a general system type and refer to a dialer entry, enter:

```
unix1200|1200 Baud dial-out to another UNIX system:

:el=^U^C^R^O^D^S^Q:ie=%$:oe=^D:tc=dial1200:
```

This entry defines a system type called unix1200. The end-of-line mark for communication with this type of remote system is ^U^C^R^O^D^S^Q. The input end-of-file mark is %$ and the output end-of-file mark is ^D. The dialer defined by the dial1200 entry is used.

3. To describe a specific system, enter:

```
zeus|CSRG ARPA VAX-11/780:pn=@:tc=unix1200:
```

This entry describes system zeus, which is described as a CSRG ARPA VAX-11. The tip command then searches the /etc/phones file for the telephone number (pn=@) and uses the attributes of a unix1200 system type (tc=unix1200).

Files

- `/etc/remote` Denotes the complete path name of the remote file.
- `/etc/phones` Lists the phone numbers used to contact remote systems.
- `/usr/lib/remote-file` Contains an example remote file.

Related Information

The `tip` command.

The `Communication with connected UNIX systems using the tip command` in Networks and communication management.

resolv.conf File Format for TCP/IP

Purpose

Defines Domain Name Protocol (DOMAIN) name-server information for local resolver routines.

Description

If the `/etc/resolv.conf` file exists, the local resolver routines either use a local name resolution database maintained by a local `named` daemon (a process) to resolve Internet names and addresses, or they use the `Domain Name Protocol` to request name resolution services from a remote DOMAIN name server host.

If no `resolv.conf` file exist than the resolver routines continue searching their direct path, which may include searching through `/etc/hosts` file or the NIS hosts map.

Note: If the `resolv.conf` file does not exist, the resolver routines attempt name resolution using the default paths, the `/etc/netsvc.conf` file, or the NSORDER environment variable.

If the host is a name server, the `resolv.conf` file must exist and contain a nameserver reference to itself as well as a default domain.

The `resolv.conf` file can contain one domain entry or one search entry, a maximum of three nameserver entries, and any number of options entries.

A domain entry tells the resolver routines which default domain name to append to names that do not end with a . (period). There can be only one domain entry. This entry is of the form:
The `DomainName` variable is the name of the local Internet domain. If there is no `domain` or `search` entry in the file, the `gethostbyname` subroutine returns the default domain (that is, everything following the first period). If the host name does not have a domain name included, the root domain is assumed.

A `search` entry defines the list of domains to search when resolving a name. Only one `domain` entry or `search` entry can be used. If the `domain` entry is used, the default search list is the default domain. A `search` entry should be used when a search list other than the default is required. The entry is of the form:

```
search DomainName ...
```

The `search` entry can have up to a maximum of 1024 character strings for the `DomainName` variable. The first `DomainName` variable is interpreted as the default domain name. The `DomainName` variable is the name of a domain that should be included in the search list.

**Notes:**
1. The `domain` entry and `search` entry are mutually exclusive. If both entries are used, the one that appears last will override the other.
2. The resolver routines require you to set the default domain. If the default domain is not set in the `/etc/resolv.conf` file, then you must set it in the `HostName` on the machine.

A `nameserver` entry defines the Internet address of a remote DOMAIN name server to the resolver routines on the local domain. This entry is of the form:

```
nameserver Address
```

The `Address` variable is the dotted decimal address of the remote name server. If more than one name server is listed, the resolver routines query each name server (in the order listed) until either the query succeeds or the maximum number of attempts have been made.

The `Address` variable is the address of the preferred network on which you want the address returned. The `Netmask` variable is the netmask of the corresponding network address.

The `options` entry specifies miscellaneous behaviors of the resolver. The entry is of the form:

```
options OptionName
```

The `OptionName` variable can have one of the following values:

- `debug` Turns on the RES_DEBUG resolver option, which enables resolver debugging.
- `ndots:n` Specifies that for a domain name with `n` or more periods (.) in it, the resolver should try to look up the domain name "as is" before applying the search list.

Entries in this file can be made using the System Management Interface Tool (SMIT), by using the `namerslv` command, or by creating and editing the file with an editor.

**Examples**

To define a domain host that is not a name server, enter:

```
domain abc.aus.century.com
nameserver 192.9.201.1
nameserver 192.9.201.2
```

The example contains entries in the `resolv.conf` file for a host that is not a name server.
Files

/usr/lpp/tcpip/samples/resolv.conf Contains the sample resolv.conf file.

Related Information
The namerslv command.
The named daemon.
The /etc/hosts file format.
The gethostbyaddr subroutine, gethostname subroutine.
TCP/IP name resolution
Name server resolution and TCP/IP name resolution in Networks and communication management.

resolv.ldap File Format for TCP/IP

Purpose
Defines Lightweight Directory Access Protocol (LDAP) server information for ldap mechanism used by local resolver subroutines.

Description
The /etc/resolv.ldap file specifies the IP address of the LDAP server, which contains the name resolution database. This database is used by the local resolver subroutines to resolve symbolic host names into Internet addresses. LDAP server specifications are obtained from resolv.ldap file only for the ldap mechanism.

Note: Although still supported, the use of the ldap mechanism is not recommended. Instead, the use of the nis_ldap mechanism is advised. For the nis_ldap mechanism, use the ldap.cfg file for configuring the LDAP server and other details.

However, if the resolv.ldap file does not exist, then the resolver subroutines continue searching their direct paths, which may include searching through a DNS server, the /etc/hosts file, or the NIS hosts map. In addition to the default paths, the resolver subroutines may also use the /etc/irs.conf file, the /etc/netsvc.conf file, or the NSORDER environment variable.

The resolv.ldap file contains one ldapserver entry, which is required, and one searchbase entry, which is optional. The ldapserver entry specifies the Internet address of the LDAP server to the resolver subroutines. The entry must take the following format:

ldapserver Address [ Port ]

The Address parameter specifies the dotted decimal address of the LDAP server. The Port parameter is optional; it specifies the port number that the LDAP server is listening on. If you do not specify the Port parameter, then it defaults to 389.

The searchbase optional entry specifies the base DN (distinguished name) of the name resolution database on the LDAP server. This entry must take the following format:

searchbase baseDN
The `baseDN` parameter specifies the starting point for the name resolution database on the LDAP server. If you do not define this entry, then the searchbase entry defaults to `cn=hosts`.

**Example**

To define an LDAP server with an IP address 192.9.201.1, that listens on the port 636, and with a searchbase `cn=hosttab`, enter the following:

```
ldapserver 192.9.201.1 636
searchbase cn=hosttab
```

**Files**

/`etc/resolv.ldap` contains the IP address of the LDAP server.

**Related Information**

The `irs.conf` file, `hosts` file format, the `netsvc.conf` file, and the `ldap.cfg` file.

TCP/IP name resolution in Networks and communication management.

---

### .rhosts File Format for TCP/IP

**Purpose**

Specifies remote users that can use a local user account on a network.

**Description**

The `$HOME/.rhosts` file defines which remote hosts (computers on a network) can invoke certain commands on the local host without supplying a password. This file is a hidden file in the local user’s home directory and must be owned by the local user. It is recommended that the permissions of the .rhosts file be set to 600 (read and write by the owner only). The group user and others should not have write permission for the .rhosts file. If write permission is granted to the group user (and others), then permission to invoke any command on the local host will not be given to the remote host. The format of the $HOME/.rhosts file is:

```
HostNameField [UserNameField]
```

When a remote command executes, the local host uses the local `/etc/hosts.equiv` file and the $HOME/.rhosts file of the local user account to validate the remote host and remote user.

**Host-Name Field**

The .rhosts file supports the following host-name entries:

```
+ HostName
-HostName
+@NetGroup
-@NetGroup
```

A + (plus sign) signifies that any host on the network is trusted. The HostName entry is the name of a remote host and signifies that any user logging in from HostName is trusted. A -HostName entry signifies that the host is not trusted. A +@NetGroup or -@NetGroup entry signifies that all hosts in the netgroup or no hosts in the netgroup, respectively, are trusted.

The @NetGroup parameter is used by Network Information Service (NIS) for grouping. Refer to the NIS netgroup file for more information.
User-Name Field
The .rhosts file supports the following user-name entries:

+ UserName
- UserName
+@NetGroup
-@NetGroup

A + (plus sign) signifies that any user on the network is trusted. The UserName entry is the login name of the remote user and signifies that the user is trusted. If no user name is specified, the remote user name must match the local user name. A -UserName entry signifies that the user is not trusted. A +@NetGroup or -@NetGroup entry signifies that all users in the netgroup or no users in the netgroup, respectively, are trusted.

The @NetGroup parameter is used by NIS for grouping. Refer to the NIS netgroup file for more information.

Examples
1. To allow remote users to log in to a local-user account, enter:
   hamlet dewey
   hamlet irving

   These entries in the local user’s $HOME/.rhosts file allow users dewey and irving at remote host hamlet to log in as the local user on the local host.

2. To prevent any user on a given remote host from logging in to a local-user account, enter:
   -hamlet

   This entry in the local user’s $HOME/.rhosts file prevents any user on remote host hamlet from logging in as a local user on the local host.

3. To allow all hosts in a netgroup to log in to a local-user account, while restricting specified users, enter:
   +@century -joe
   +@century -mary
   +@century

   This entry in the local user’s $HOME/.rhosts file allows all hosts in the century netgroup to log in to the local host. However, users joe and mary are not trusted, and therefore are requested to supply a password. The deny, or - (minus sign), statements must precede the accept, or + (plus sign), statements in the list. The @ (at sign) signifies the network is using NIS grouping.

Files
/etc/hosts.equiv
   Specifies remote systems that can execute commands on the local system.
   netgroup
   Lists the groups of users on the network.

Related Information
The lpd command, rcp command, rdist command, rdump command, rlogin command, rsh command, ruser command.

The NIS netgroup file.

The rlogind daemon, rshd daemon.

The TCP/IP /etc/hosts.equiv file format.
TCP/IP name resolution in *Networks and communication management.*

---

**sccsfile File Format**

**Purpose**
Describes the format of a Source Code Control System (SCCS) file.

**Description**
The SCCS file is an ASCII file consisting of the following logical parts:

- **Checksum**
  The logical sum of all characters except the characters in the first line

- **Delta table**
  Information about each delta including type, SCCS identification (SID) number, date and time of creation, and comments about the delta

- **User Names**
  Login names, group names, or numerical group IDs of users who are allowed to add or remove deltas from the SCCS file

- **Header flags**
  Flags defining how some SCCS commands work with the SCCS file, or defining values for identification keywords in the file

- **Comments**
  Descriptive information about the file

- **Body**
  The actual text lines intermixed with control lines

**Note:** Several lines in an SCCS file begin with the ASCII SOH (start-of-heading) character (octal 001). This character is called the control character and is represented graphically as the @ (at sign) in the following text. Any line described in the following text that does not begin with the control character contains text from the source file. Text lines cannot begin with the control character.

**Checksum**
The checksum is the first line of an SCCS file. This line has the following format:

@hNumber

The control character and variables in the checksum line have the following meanings:

- **@h** Designates a magic number of 064001 octal (or 0x6801).
- **Number** Represents the logical sum of all characters in the SCCS file (not including the characters in this line). It is recalculated each time the SCCS file is updated with SCCS commands, and is used to detect possibly damaging changes made to an SCCS file by non-SCCS commands.

**Delta Table**
Each time a group of changes, known as a delta, is made to an SCCS file, the delta table creates a new entry. Each entry contains descriptive information about the delta. The @s (at sign, letter s) character defines the beginning of a delta table entry, and the @e (at sign, letter e) character defines the end of the entry. For each delta created, there is a delta table entry in the following format:

@s NumberLinesInserted/NumberLinesDeleted/NumberLinesUnchanged
@d DeltaType SID Date Time UserID Number PreNumber
@t NumbersIncluded . . .
@x NumbersExcluded . . .
@g NumbersIgnored . . .
@m ModificationRequestNumber
@c Comments . . .

The control characters and variables in the delta table entries have the following meanings:

- **@s** Designates the first line of each entry, which contains the number of lines inserted, deleted, and unchanged from the previous delta.
Designates the second line of each entry, which contains the following variables:

**Delta Type**
Type of delta. The letter d designates a normal delta; the letter r designates a delta that has been removed with the `rmdel` command.

**SID**
SCCS ID (SID) of the delta.

**Date**
Date, in the YY/MM/DD format, that the delta was created.

**Time**
Time, in the HH:MM:SS format, that the delta was created.

**UserID**
Login name that corresponds to the real user ID at the time the delta was created.

**Number**
Serial numbers of the delta.

**PreNumber**
Serial numbers of the delta’s predecessor.

**i**
Indicates the serial numbers of the deltas that are included in the creation of this delta by using the `get` command with the `-i` flag. This line can contain several delta numbers and is optional.

**x**
Indicates the serial numbers of the deltas that were excluded from the creation of this delta by using the `get` command with the `-x` flag. This line can contain several delta numbers and is optional.

**g**
Indicates the serial numbers of the deltas that were ignored in the creation of this delta by using the `delta` command with the `-g` flag. This line can contain several delta numbers and is optional.

**m**
Indicates a modification request (MR) number associated with the delta. There can be several MR lines in an SCCS file, each one containing a different MR number. These lines are optional.

**c**
Comment lines associated with the delta. There can be several comment lines in an SCCS file. These lines are optional.

**e**
Ends the delta table entry.

**User Names**
This section of the file contains the list of login names, group names, or numerical group IDs of users who can add deltas to the file. The names and IDs are separated by new-line characters. This section uses the following control characters:

**u**
A bracketing line that indicates the beginning of a user-name list. This line appears before the first line in the list.

**U**
A bracketing line that indicates the end of a user name list. This line appears after the last line in the list.

An empty list allows any user to make a delta. The list is changed using the `admin` command with the `-a` or `-e` flag.

**Header Flags**
Flags control commands and define keywords used internally in the SCCS. Header flags are set using the `admin` command with various flags. The format of each line is:

`@f Flag Text`

The control character and variables in the header flags section have the following meanings:

**fb**
Branch. Allows the use of the `-b` flag of the `get` command to cause a branch in the delta tree.

**fc**
Ceiling. Defines the highest release number from 0 through 9999 that can be retrieved by a `get` command for editing. This release number is called the ceiling release number.

**fd**
Default SCCS ID. Defines the default SID to be used when one is not specified with a `get` command. When this flag is not set, the `get` command uses the most recently created delta.

**ff**
Floor. Defines the lowest release number from 0 through 9999 that can be retrieved by a `get` command for editing. This release number is called the floor release number.
ID keywords. Controls the No ID keywords error warning message. When this flag is not set, the message is only a warning. When this flag is set, the absence of ID keywords causes an error and the delta fails.

Joint edit. Causes the get command to allow concurrent edits of the same base SID.

Lock releases. Defines a list of releases that cannot be edited with the get command using the -e flag.

Module name. Defines the replacement of a module name for the 11 identification keyword. This value is used to override the default.

No changes. Causes the delta command to insert null deltas (delta entries with no changes) for any skipped releases when a delta for a new release is created. For example, delta 5.1 is created after delta 2.1, skipping releases 3 and 4. When this flag is omitted, skipped releases are omitted from the delta table.

User-defined flag. Defines the replacement of the identification keyword.

Type of program. Defines the replacement of the identification keyword.

Program name. Controls prompting for MR numbers in addition to comments on delta creation. If a value is assigned, it defines an MR number validity-checking program.

Comments
When comments are taken from a file containing descriptive text by using the admin command with the -t flag option, the contents of that file are displayed in the comments section. Typically, the comments section contains a description of the purpose of the entire file and uses the following control characters:

A bracketing line that indicates the beginning of the comments section. This line appears before the first comment line.

A bracketing line that indicates the end of the comments section. This line appears after the last comment line.

Body
The body consists of two types of lines: control lines and text lines. Control lines bracket text lines. The text lines contain pieces of text that were inserted or deleted for a particular version of the file. The control lines that bracket a piece of text indicate whether a piece of text was inserted or deleted, and in what version. When a particular version of a file is created from the SCCS file, the control lines identify the pieces of text that should be added or deleted for that version of the file.

Control lines can be nested within one another, so the same portion of text can be bracketed by several sets of control lines. The body of a long SCCS file can be very complicated. The SCCS commands, however, provide a better way to understand the different versions of an SCCS file.

Indicates an insert control line. The Number variable indicates the serial number that corresponds to the delta for the control line. Text inserted between this control line and an end control line with the same serial number was inserted as part of the delta that corresponded to the same serial number.

Indicates a delete control line. The Number variable indicates the serial number that corresponds to the delta for the control line is indicated by the Number variable. Text deleted between this control line and an end control line with the same serial number was deleted as part of the delta that corresponded to the same serial number.

Indicates an end control line. The serial number that corresponds to the delta for the control line is indicated by the Number variable. This indicates the end of a section of text to be inserted or deleted.

Within the text are also identification keywords that are specific to the SCCS file system. These keywords represent identifying information about the SCCS file. When using the get command without the -e or -k flag, these keywords will be replaced by their values. Because different versions have different identifying
information, the identification keywords provide an easy way for the SCCS file system to provide the correct identifying information for any version of the file requested by the `get` command. Keywords can be used to provide several kinds of information:

- **Version identification information:**

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>%M%</td>
<td>Module name; the value of the <code>m</code> header flag in the SCCS file</td>
</tr>
<tr>
<td>%I%</td>
<td>SID (1, 1, 0, 0)</td>
</tr>
<tr>
<td>%R%</td>
<td>Release</td>
</tr>
<tr>
<td>%L%</td>
<td>Level</td>
</tr>
<tr>
<td>%B%</td>
<td>Branch</td>
</tr>
<tr>
<td>%S%</td>
<td>Sequence</td>
</tr>
</tbody>
</table>

- **Time and date information:**

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>01/09/28</td>
<td>Date of the current <code>get</code> command (YY/MM/DD)</td>
</tr>
<tr>
<td>9/28/01</td>
<td>Date of the current <code>get</code> command (MM/DD/YY)</td>
</tr>
<tr>
<td>14:33:53</td>
<td>Time of the current <code>get</code> command (HH:MM:SS)</td>
</tr>
<tr>
<td>00/12/05</td>
<td>Date newest applied delta was created (YY/MM/DD)</td>
</tr>
<tr>
<td>12/5/00</td>
<td>Date newest applied delta was created (MM/DD/YY)</td>
</tr>
<tr>
<td>14:59:04</td>
<td>Time newest applied delta was created (HH:MM:SS)</td>
</tr>
</tbody>
</table>

- **Name information:**

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>/family/aix/vc/8/9/7/3/s.11</td>
<td>SCCS file name</td>
</tr>
<tr>
<td>/family/aix/vc/8/9/7/3/s.11</td>
<td>Full path name of the SCCS file</td>
</tr>
</tbody>
</table>

- **Flag values:**

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>-q</td>
<td>Value of the <code>-q</code> header flag in the SCCS file.</td>
</tr>
<tr>
<td>-t</td>
<td>Module type; the value of the <code>-t</code> header flag in the SCCS file.</td>
</tr>
</tbody>
</table>

- **Line numbers:**

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>562</td>
<td>The current line number. This keyword identifies message output by the program. It should not be used on every line to provide sequence numbers.</td>
</tr>
</tbody>
</table>

- **Constructing `what` strings:**

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>@(#) 11 1.1@(#)</td>
<td>A shorthand notation for constructing <code>what</code> strings for program files specific to other operating systems. Its value equals the following key letters:</td>
</tr>
<tr>
<td></td>
<td><code>@(#) 11 1.1@(#) = @(#) 11 1.1@(#)</code></td>
</tr>
<tr>
<td><code>src/idd/en_US/files/aixfiles/sccsfile.ide, idaixfiles, idd520</code></td>
<td>A shorthand notation for constructing <code>what</code> strings for program files specific to this operating system. Its value is the characters and key letters:</td>
</tr>
<tr>
<td></td>
<td><code>src/idd/en_US/files/aixfiles/sccsfile.ide, idaixfiles, idd520 = @(#)11&lt;horizontal-tab&gt;1.1</code></td>
</tr>
<tr>
<td>@(#)</td>
<td>The 4-character string <code>@(#)</code> (at sign, left parenthesis, pound sign, right parenthesis) recognized by the <code>what</code> command.</td>
</tr>
</tbody>
</table>
Related Information
Source Code Control System (SCCS) Overview in AIX 5L Version 5.3 General Programming Concepts: Writing and Debugging Programs contains general information about the SCCS file system.

The `admin` command, `delta` command, `get` command, `prs` command, `rmdel` command, `what` command.

services File Format for TCP/IP

Purpose
Defines the sockets and protocols used for Internet services.

Description
The `/etc/services` file contains information about the known services used in the DARPA Internet network. Each service is listed on a single line corresponding to the form:

```
ServiceName PortNumber/ProtocolName Aliases
```

These fields contain the following information:

- **ServiceName**: Specifies an official Internet service name.
- **PortNumber**: Specifies the socket port number used for the service.
- **ProtocolName**: Specifies the transport protocol used for the service.
- **Aliases**: Specifies a list of unofficial service names.

Items on a line are separated by spaces or tabs. Comments begin with a `#` (pound sign) and continue until the end of the line.

If you edit the `/etc/services` file, run the `refresh -s inetd` or `kill -1 InetdPID` command to inform the `inetd` daemon of the changes.

Examples
Entries in the `services` file for the `inetd` internal services may look like this:

```
echo    7/tcp
echo    7/udp
discard 9/tcp    sink null
discard 9/udp    sink null
daytime 13/tcp
daytime 13/udp
chargen 19/tcp   ttys source
chargen 19/udp   ttys source
ftp      21/tcp
time    37/tcp    timeserver
time    37/udp    timeserver
```

Related Information
The `getservent` subroutine.

The `/etc/inetd.conf` file format.

Object Data Manager (ODM) Overview for Programmers in AIX 5L Version 5.3 General Programming Concepts: Writing and Debugging Programs.

TCP/IP daemons in Networks and communication management.
**setmaps File Format**

**Purpose**
Defines the text of a code-set map file and a terminal map file.

**Description**
The text of a code set map file consists of a description of the code set. The text of a terminal map file consists of a set of rules.

**Code-Set Map File**
The text of a code set map file is a description of the code set. It specifies the optional converter modules to push on the stream. The code set map file is located in the `/usr/lib/nls/csmap` directory. Its name is the code set name.

The code set map file contains the following lines:

<table>
<thead>
<tr>
<th>Name</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>M</td>
</tr>
<tr>
<td>Multibyte handling</td>
<td>EUC</td>
</tr>
<tr>
<td>ioctl EUC_WSET</td>
<td>w1:d1, w2:d2, w3:d3</td>
</tr>
<tr>
<td>lower converter</td>
<td>/usr/lib/drivers/lwconv</td>
</tr>
<tr>
<td>upper converter</td>
<td>/usr/lib/drivers/upconv</td>
</tr>
</tbody>
</table>

The lines have the following meaning:

- **Name**: Specifies the code set name. It is also the code set map file name.
- **Type**: Specifies the code set type. It can be one of the following:
  - M: Denotes a multibyte code set.
  - S: Denotes a single byte code set.
- **Multibyte handling**: Specifies the type of multibyte handling of the code set. This line is required only if **Type** is M. It must be EUC, denoting an EUC multibyte code set.
- **ioctl EUC_WSET**: Specifies the parameters for the EUC_WSET ioctl operation. This line is required only if **Type** is M. The `w1, w2, and w3` parameters specify the memory width of the code set; the `d1, d2, and d3` parameters specify the screen width of the code set.
- **lower converter**: Specifies the lower and upper converters to use on the stream. This line is required only if the code set is a non-EUC multibyte code set.
- **upper converter**: Specifies the lower and upper converters to use on the stream. This line is required only if the code set is a non-EUC multibyte code set.

For example, the code set map file for the ISO 8859-1 code set would contain the following lines:

```plaintext
Name: ISO8859-1
Type: S
```

Another example: the code set map file for the IBM-943 code set would contain the following lines:

```plaintext
Name: IBM-943
Type: M
Multibyte handling: EUC
ioctl EUC_WSET: 2:2,1:1,2:2
lower converter: /usr/lib/drivers/lc_sjis
upper converter: /usr/lib/drivers/up_sjis
```
**Terminal Map File**
The text of a terminal map file is a set of rules. Each rule has the following format:

\[ \text{pattern:replacement} \]

The size of the input pattern string is limited to 10 characters in length and the size of the replacement pattern string is limited to 16 characters in length.

The pattern string can include the following special characters:

- `?` Matches any single byte.
- `@x` Matches this rule only if the pattern processor is in state \( x \), where \( x \) is any single byte. (This sequence does not match a character in the input buffer.)
- `\?`, `\@`, or `\` Prevents the pattern processor from interpreting `?` (question mark), `@` (at sign), or `\` (backslash) as special characters.
- `\ddd` Represents any byte in octal notation.
- `\xdd` Represents any byte in hexadecimal notation.

The replacement string can include the following special characters:

- `$n` Uses the \( n \)th character in the input string that matched this pattern, where \( n \) is a decimal digit.
- `@x` Moves the pattern processor into state \( x \). (This sequence does not become part of the replacement string.)
- `\$`, `\@`, or `\` Prevents the pattern processor from interpreting $, @, or \ as special characters.
- `\ddd` Represents any byte in octal notation.
- `\xdd` Represents any byte in hexadecimal notation.

**Files**

- `/usr/lib/nls/cmap/sbcs` Code set map for a single-byte code page
- `/usr/lib/nls/cmap/IBM-932` Code set map for the IBM-932 code page
- `/usr/lib/nls/cmap/IBM-943` Code set map for the IBM-943 code page
- `/usr/lib/nls/cmap/IBM-eucTW` Code set map for the IBM-eucTW code page
- `/usr/lib/nls/termmap/*.in` Input map files
- `/usr/lib/nls/termmap/*.out` Output map files

**Related Information**
The `eucioctl.h` file.
The `setmap` command.
The `setsmap` subroutine.
`tty Subsystem Overview` in *AIX 5L Version 5.3 General Programming Concepts: Writing and Debugging Programs*.

**simprof File Format**

**Purpose**
Specifies PC Simulator startup options.
Description

When you start PC Simulator with the `pcsim` command, PC Simulator searches for a profile of startup options. The profile used by PC Simulator is the `simprof` file format. It is a pure ASCII text file that you can edit with any text editor.

You can specify the name of a profile with the `-profile` flag at the `pcsim` command. If you do not enter a `-profile` flag, PC Simulator searches for the `simprof` default profile. This sample profile, included with PC Simulator, is located in the `/usr/lpp/pcsim/samples` directory.

You can define more than one profile. These profiles can be for different users or for starting PC Simulator with different options. PC Simulator first searches for the specified profile in the current working directory, then in the `$HOME` directory, and finally in the `/usr/lpp/pcsim` directory. To operate with only one profile, you can copy the `simprof` sample profile to one of these directories, and edit it to set the options you want.

Even if PC Simulator finds a profile, it searches all three directories. It can, therefore, find more than one profile with the same file name. If this happens, PC Simulator accumulates options from each profile. It overlays values for the same option in each profile and uses the last value it reads. You can set options with flags from the command line that override any options in a profile.

Examples

A simulator profile resembles an AIXwindows default profile. Options are listed by flag name, followed by a : (colon), then a parameter value. The `simprof` sample profile included with PC Simulator is similar to this example, except that it includes no parameter values.

If an option is not listed or no value is specified, PC Simulator starts with the default value for this option. A blank space between the colon and parameter value is optional. Any text following a # (pound sign) is a comment. PC Simulator expands environment variables inside the `simprof` file.

Note: If there is no diskette drive present, the entries for `Adiskette` and `Bdiskette` should be removed from the profile. If there is only one diskette drive present, the entry for `Bdiskette` should be removed from the profile.

```
Cdrive : /home/dos1/txt.fil  # select file /home/dos1/txt.fil
      # for fixed disk C:
Ddrive : /home/dos2        # select directory /home/dos2
      # for fixed disk D:
permission : 666           # read/write permissions to
      # all users for files saved
      # to fixed disk
Adiskette : 3               # select 3.5-inch diskette drive
Bdiskette :                 # no B diskette drive selected
dtime : 5                   # release diskette drive to
      # AIX after 5 seconds
display :                   # use default AIXwindows
      # server, unix:0
dmode : V                    # select VGA display mode
geometry :                  # use default window size
      # & position, 720x494+152+265
iconGeometry : =64X64+10+10  # size and position of icon
iconName :                  # use default, pcsim
kbdmap :                    # no file selected
name : BUDGET               # name in window title bar
refresh : 100               # refresh display every
      # 100 milliseconds
lpt1 : lp0                  # emulate DOS lpt1 with AIX lp0
lpt2 :                       # none selected
lpt3 :                       # none selected
mouse : com1                # emulate Microsoft serial mouse
```
ptime : 30        # print job file buffering
         # time out after 30 seconds
xmemory : 1024    # provide 1MB extended memory

Files

/usr/lpp/pcsim/samples/simprof   Contains an example startup profile.

Standard Resource Record Format for TCP/IP

Purpose
Defines the format of lines in the named data files.

Description
Records in the named files are called resource records. Files using the standard resource record format are:

- DOMAIN data file
- DOMAIN reverse data file
- DOMAIN cache file
- DOMAIN local file

Resource records in the named files have the following general format:

{Name} {TTL} AddressClass RecordType RecordSpecificData

Field Definitions

Name

Varies depending on the RecordType field. The Name field can specify the name of a domain, a zone of authority, the name of a host, the alias of a host or of a mailbox, or a user login ID. The Name field must begin in column one. If this field is left blank, the name defaults to the value of the previous resource record.

TTL

Time to live. This specifies how long the record is stored in the database. If this field is left blank, the time to live defaults to the time to live specified in the start of authority record. This field is optional.

AddressClass

Address class of the record. There are three valid entries for this field: ANY for all address classes, IN for Internet, and CHAOS for Chaos net.

RecordType

The type of resource record. Valid record types are:

- SOA: Start of authority record
- NS: Name server record
- A: Address record
- HINFO: Host information record
- WKS: Well-known services record
- CNAME: Canonical name record
- PTR: Domain name pointer record
- MB: Mailbox record
- MR: Mail rename name record
- MINFO: Mailbox information record
- MG: Mail group member record
- MX: Mail exchanger record

Details and examples of record types are given below.

RecordSpecificData

These fields are dependent on the RecordType field.
Although case distinctions are kept when loading databases, all queries to the name server database are case insensitive.

The following characters have special meanings:

**Special Characters**

- **.**
  - If used in the *Name* field, a . (period) indicates the current domain.
  - **Note:** Use the . (period) at the end of resource records to append the path of the current domain.

- **..**
  - If used in the *Name* field, two periods indicate the null domain name of the root domain.

- **@**
  - If used in the *Name* field, an @ (at sign) indicates the current origin.

- **\X**
  - Where X is any character except numbers 0 through 9 or the character . (period), a backslash preceding a character indicates that the character's special meaning should not be used. For example, \@ (backslash, at sign) can be used to put an @ character in the label of an entry in the *Name* field.

- **\DDD**
  - Where each D is any number between 0 and 9. Each number is identified as the binary octet corresponding to the number. These octets are not checked for special meaning.
  - **Note:** The \DDD character is not used in the *Name* field of a resource record.

- **( )**
  - Parentheses indicate that data broken into more than one line should be grouped together. The () (parentheses) are currently used in the SOA and WKS resource records.

- **;**
  - Indicates a comment line. All characters after the ; (semicolon) are ignored.

- *** (asterisk)**
  - An * (asterisk) indicates wildcards.
  - **Note:** The * (asterisk) character is not used in the *Name* field of a resource record.

There are two special types of lines that are not data lines. Instead they specify special processing. These lines are the $INCLUDE and $ORIGIN lines.

**Special Types of Lines**

- **$INCLUDE FileName**
  - This line begins in column one and is followed by a file name. It indicates that the specified file should be included in the name server database. This is useful in separating different types of data into multiple files. For example:
    
    $INCLUDE /usr/named/data/mailbox

    indicates that this file should be loaded into the name server's database. Data files specified by the $INCLUDE line are not treated differently from any other named data file.

- **$ORIGIN OriginName**
  - This line begins in column one and is followed by the name of a domain. This line indicates that the origin from more than one domain in a data file should be changed.

**Resource Record Types**

Following is a list of the resource record types used in the named data files:

- Start of authority record
- **Name server record**
- Address record
The start of authority (SOA) record indicates the start of a zone of authority. There should be only one start of authority record per zone, indicated by a value of SOA in the RecordType field. However, the SOA record for the zone should be in each named.data and named.rev file on each name server in the zone. Its structure corresponds to the following format:

{Name}{TTL} AddressClass RecordType Origin PersonInCharge
@ IN SOA merl.century.com jane.merl.century.com
(1.1 ;Serial
3600 ;Refresh
600 ;Retry
3600000 ;Expire
86400) ;Minimum

**Fields:**

- **Name**: Name of the zone.
- **TTL**: Time to live.
- **AddressClass**: Internet (IN).
- **RecordType**: Start of authority (SOA).
- **Origin**: Name of the host on which this data file resides.
- **PersonInCharge**: Person responsible for keeping the data file current. The format is similar to a mailing address, but the @ (at sign) that normally separates the user from the host name is replaced by a . (period).
- **Serial**: Version number of this data file. This number should be incremented each time a change is made to the data. The upper limit for the number to the right of the decimal point is 9999.
- **Refresh**: The number of seconds after which a secondary name server checks with the primary name server to see if an update is needed. A suggested value for this field is 3600 (1 hour).
- **Retry**: The number of seconds after which a secondary name server is to retry after a refresh attempt fails. A suggested value for this field is 600 (10 minutes).
- **Expire**: The upper limit in seconds that a secondary name server can use the data before it expires because it has not been refreshed. This value should be fairly large, and a suggested value is 3600000 (42 days).
- **Minimum**: The minimum time, in seconds, to use as time-to-live values in resource records. A suggested value is 86400 (one day).
Name Server Record

The name server record specifies the name server responsible for a given domain. There should be one name server record for each primary server for the domain, indicated by a value of NS in the RecordType field. The name server record can be in the named.data file, the named.rev file, the named.ca file, and the named.local file. Its structure corresponds to the following format:

**{Name} {TTL} AddressClass RecordType NameServerName**

```
IN   NS   arthur.century.com
```

**Fields:**

- **Name**: Indicates the domain serviced by the specified name server. In this case, the domain defaults to the value in the previous resource record.
- **TTL**: Time to live.
- **AddressClass**: Internet (IN).
- **RecordType**: Name server (NS).
- **NameServerName**: The name server responsible for the specified domain.

Address Record

The address record specifies the address for the host and is indicated by a value of A in the RecordType field. Address records can be entries in the named.ca, named.data, and named.rev files. Its structure corresponds to the following format:

**{Name} {TTL} AddressClass RecordType Address**

```
arthur IN A 132.10.8.1
         IN A 10.0.4.1
```

**Fields:**

- **Name**: Name of the host.
- **TTL**: Time to live.
- **AddressClass**: Internet (IN).
- **RecordType**: Address (A).
- **Address**: Internet address of the host in dotted decimal form. There should be one address record for each Internet address of the host.

If the name server host for a particular domain resides inside the domain, then an A (address) resource record that specifies the address of the server is required. This address record is only needed in the server delegating the domain, not in the domain itself. If, for example, the server for domain aus.century.com was fran.aus.century.com, then the NS record and the required A record would look like:

```
fran.aus.century.com. IN A 192.9.201.14
```

Host Information Record

The host information (HINFO) record lists host specific information, and is indicated by HINFO in the RecordType field. This lists the hardware and operating system that are running at the specified host. Note that the hardware and operating system information is separated by a single space. There must be one host information record for each host. The HINFO record is a valid entry in the named.data and the named.rev files. Its structure corresponds to the following format:

**{Name} {TTL} AddressClass RecordType Hardware OS**
Fields:

Name               Name of the host.
TTL                Time to live.
AddressClass       Address class. Valid values are IN for Internet and CHAOS for Chaos net.
RecordType         Host information (HINFO).
Hardware           Make and model of hardware.
OS                  Name of the operating system running on the host.

Well-Known Services Record

The well-known services (WKS) record lists the well-known services supported by a particular protocol at a specified address. This record is indicated by WKS in the RecordType field. Although TCP/IP provides the record for backward compatibility, it is now obsolete.

The services and port numbers come from the list of services in the /etc/services file. There should be only one WKS record per protocol per address. The WKS record is a valid entry in the named.data file. Its structure corresponds to the following format:

{Name}{TTL} AddressClass RecordType Address Protocol ListOfServices
  IN    WKS  125.10.0.4  UDP  (who route timed domain)
  IN    WKS  125.10.0.4  TCP  (echo telnet ftp netstat finger)

Fields:

Name               Name of the host. In this case, the name of the host defaults to the value in the previous resource record.
TTL                Time to live
AddressClass       Internet (IN)
RecordType         Well-known services (WKS)
Address            Internet address of the adapter in dotted decimal form
Protocol           Protocol used by the list of services at the specified address
ListOfServices     Services supported by a protocol at the specified address

Canonical Name Record

The canonical name record specifies an alias for a canonical name (CNAME), and is indicated by CNAME in the RecordType field. The CNAME record is the only Resource record that can use the alias of a canonical name. All other resource records must use the full canonical (or domain) name. The CNAME record is a valid entry in the named.data file. For each CNAME record, there must be a corresponding address (A) record. Its structure corresponds to the following format:

{Aliases} {TTL} AddressClass RecordType CanonicalName
  knight   IN    CNAME    lancelot
  john     IN    CNAME    lancelot

Fields:

Aliases           Alias by which the host is known
TTL               Time to live
AddressClass      Internet (IN)
RecordType        Canonical name (CNAME)
CanonicalName     Official name associated with the alias
IN-ADDR.ARPA Record

The structure of names in the domain system is set up in a hierarchical fashion. The address of a name can be found by tracing down the domain structure, contacting a server for each label in the name. Because the structure is based on names, there is no easy way to translate a host address back into its host name.

In order to allow simple reverse translation, the IN-ADDR.ARPA domain was created. This domain uses host addresses as part of a name that points to the data for that host. The IN-ADDR.ARPA domain provides an index to the resource records of each host based on its address. There are subdomains within the IN-ADDR.ARPA domain for each network, based on network number. Also, to maintain consistency and natural groupings, the 4 octets of a host number are reversed. The IN-ADDR.ARPA domain is defined by the IN-ADDR.ARPA record in the named.boot files and the DOMAIN hosts data file.

For example, the ARPANET is net 10, which means that there is a domain called 10.in-addr.arpa. Within this domain, there is a PTR resource record at 51.0.0.10.IN-ADDR, which points to the resource records for the host sri-nic.arpa (whose address is 10.0.0.51). Since the NIC is also on the MILNET (net 26, address 26.0.0.73), there is also a PTR resource record at 73.0.0.26.in-addr.arpa that points to the same resource records for SRI-NIC.ARPA. The format of these special pointers is defined in the following section on PTR resource records, along with the examples for the NIC.

Domain-Name Pointer Record

The Domain-Name Pointer record allows special names to point to some other location in the domain. This record is indicated by PTR in the RecordType field. PTR resource records are mainly used in IN-ADDR.ARPA records to translate addresses to names.

Note: PTR records should use official host names, not aliases.

The PTR record is a valid entry in the named.rev file. Its structure corresponds to the following format:

{Aliases} {TTL} AddressClass RecordType RealName

7.0 IN PTR arthur.century.com.

Fields:

Aliases Specifies where this record should point in the domain. Also specifies the Internet address of the host with the octets in reverse order. If you have not defined the IN-ADDR.ARPA domain in your named.boot file, this address must be followed by .in-addr.arpa.

TTL Time to live.

AddressClass Internet (IN).

RecordType Pointer (PTR).

RealName The domain name of the host to which this record points.

Gateway PTR Record

The IN-ADDR domain is also used to locate gateways on a particular network. Gateways have the same kind of PTR resource records as hosts, but they also have other PTR records used to locate them by network number alone. These records have 1, 2, or 3 octets as part of the name, depending on whether they are class A, B, or C networks, respectively.

The gateway host named gw, for example, connects three different networks, one for each class, A, B, and C. The gw gateway has the standard resource records for a host in the csl.sri.com zone:

gw.csl.sri.com. IN A 10.2.0.2
IN A 128.18.1.1
IN A 192.12.33.2

Chapter 2. File Formats 575
In addition, this gateway has one of the following pairs of number-to-name translation pointers and gateway location pointers in each of the three different zones (one for each network). In each example, the number-to-name pointer is listed first, followed by the gateway location pointer.

**Class A**

2.0.2.10.in-addr.arpa. IN PTR gw.csl.sri.com.

**Class B**

1.1.18.128.in-addr.arpa. IN PTR gw.csl.sri.com.
18.128.in-addr.arpa. IN PTR gw.csl.sri.com.

**Class C**


For example, a user named elizabeth used the following resource record to have her mail delivered to host venus.abc.aus.century.com:

elizabeth IN MB venus.abc.aus.century.com.

**Mailbox Record**

The mailbox (MB) record defines the machine where a user wants to receive mail, and is indicated by MB in the RecordType field. The MB record is a valid entry in the named.data file. Its structure corresponds to the following format:

{Aliases} {TTL} AddressClass RecordType Machine

Fields:

- **Aliases**: The user login ID
- **TTL**: Time to live
- **AddressClass**: Internet (IN)
- **RecordType**: Mailbox (MB)
- **Machine**: Name of the machine at which the user wants to receive mail

**Mail Rename Name Record**

The mail rename (MR) name record allows a user to receive mail addressed to a list of aliases. This record is indicated by MR in the RecordType field. The MR record is a valid entry in the named.data file. Its structure corresponds to the following format:

{Aliases} {TTL} AddressClass RecordType CorrespondingMB

Fields:

- **Aliases**: Alias for the mailbox name listed in the last field.
- **TTL**: Time to live.
- **AddressClass**: Internet (IN).
- **RecordType**: Mail rename (MR).
- **CorrespondingMB**: The name of the mailbox. This record should have a corresponding MB record.
Mailbox Information Record
The mailbox information (MINFO) record creates a mail group for a mailing list, and is indicated by MINFO in the RecordType field. This record usually has a corresponding mail group record, but may also be used with a mailbox record. The MINFO record is a valid entry in the named.data file. Its structure corresponds to the following format:

{Name} {TTL} AddressClass RecordType Requests Maintainer
postmaster IN MINFO post-request greg.century.com

Fields:
- Name: The name of the mailbox.
- TTL: Time to live.
- AddressClass: Internet (IN).
- RecordType: Mail Information record (MINFO).
- Requests: Where mail requests (such as a request to be added to the mailing list) should be sent.
- Maintainer: The mailbox that should receive error messages. This is particularly useful when mail errors should be reported to someone other than the sender.

Mail Group Member Record
The mail group member (MG) record lists the members of a mail group. This record is indicated by MG in the RecordType field. The MG record is a valid entry in the named.data file. Its structure corresponds to the following format:

{MailGroupName} {TTL} AddressClass RecordType MemberName
depth IN MG Tom

Fields:
- MailGroupName: Name of the mail group.
- TTL: Time to live.
- AddressClass: Internet (IN).
- RecordType: Mail group member record (MG).
- MemberName: The login ID of the group member.

Mail Exchanger Record
The mail exchanger (MX) records identify machines (gateways) that know how to deliver mail to a machine that is not directly connected to the network. This record is indicated by MX in the RecordType field. Wildcard names containing an * (asterisk) can be used for mail routing with MX records. There may be servers on the network that state that any mail to a domain is to be routed through a relay. The MX record is a valid entry in the named.data file. Its structure corresponds to the following format:

{Name} {TTL} AddressClass RecordType PrefValue MailExchanger
Ann.bus.com IN MX 0 Hamlet.Century.Com
*.dev.bus.com IN MX 0 Lear.Century.Com

Fields:
- Name: Specifies the full name of the host to which the mail exchanger knows how to deliver mail.
  Note: The * (asterisk) in the second name entry is a wildcard name entry. It indicates that any mail to the domain dev.bus.com should be routed through the mail gateway Lear.Century.Com.
- TTL: Time to live.
AddressClass Internet (IN).
RecordType Mail Exchanger (MX).
PrefValue Indicates the order the mailer should follow when there is more than one way to deliver mail to a host.
MailerExchanger The full name of the mail gateway. See RFC 974 for more information.

Examples
The following is an example of a mailing list:

depth IN MINFO dept-request jane.merlin.century.com
IN MG greg.arthur.century.com
IN MG tom.lancelot.century.com
IN MG gary.guinevere.century.com
IN MG kent.gawain.century.com

Related Information
The named daemon.

The DOMAIN Data file format, DOMAIN Cache file format, DOMAIN Local file format, DOMAIN Reverse file format.
TCP/IP name resolution in Networks and communication management.
Name server resolution in AIX 5L Version 5.3 Communications Programming Concepts.

Sysfiles File Format for BNU

Purpose
Gives system administrators flexibility in configuring their Systems, Devices and Dialers files for use with BNU commands.

Description
The /etc/uucp/Sysfiles file let system administrators specify alternate Systems, Devices and Dialers files to replace the default files in the /etc/uucp directory or to supplement those files to enable a separation of the data needed to access remote systems. It is organized so a user can invoke two distinct types of services, uucico and cu. The uucico service refers to the /usr/sbin/uucp/uucico command and the commands that invoke it, for example uucp, uux, uusend, uucico. It automatically logs into remote systems and sends and receives data. The cu service connects to remote systems without attempting to login and uses the cu, ct, and slattach commands to contact remote systems. The responses to the user name and password prompts as well as any data transfer is the responsibility of the user. Based upon these differences a system administrator can split the data used to contact remote systems according the service types

The Sysfiles file contains a description of each BNU service on the local system that can establish a remote connection. Each line in the Sysfiles file corresponds to the following syntax:

```
service=uucico|cu [systems=filename[:filename]] \ 
[devices=filename[:filename]] \ 
[dialers=filename[:filename]] 
```

If a service does not have a corresponding line in the Sysfiles file, the default files are used.
Examples

1. A **Sysfiles** configuration that splits the configuration files for **uucico** and **cu** into different sets of files would be as follows:

   ```
   service=uucico systems=Systems.cico devices=Devices.cico
dialers=Dialers.cico
   service=cu systems=Systems.cu devices=Devices.cu
dialers=Dialers.cu
   ```

   These two lines in a **Sysfiles** file state that two separate sets of **Systems**, **Devices** and **Dialers** files are used for each service. Each service is specified by the `service=` at the beginning of a line with no leading white space. The files used for each service is named on the same line according to the substrings appended to the `systems=`, `devices=` and `dialers=`. Their default location is in the `/etc/uucp` directory.

2. A configuration to split the **uucico** and **cu** service entries into separate files, but to combine common configuration data would be as follows:

   ```
   service=uucico systems=Systems.cico:Systems
devices=Devices.cico:Devices
dialers=Dialers.cico:Dialers
   service=cu systems=Systems.cu:Systems
devices=Devices.cu:Devices
dialers=Dialers.cu:Dialers
   ```

   This example provides separate **Systems**, **Devices**, and **Dialers** files for each service, but combines any common data into the default files. As the example shows, multiple **Systems**, **Devices** and **Dialers** files can be specified for each service. A colon is used as the filename delimiter in such a case.

3. This example specifies separate **Systems** files for each service. Each service uses the default **Devices** and **Dialers** files.

   ```
   service=uucico systems=Systems.cico
   service=cu systems=Systems.cu
   ```

   If no **Sysfiles** service entry is made for a **Systems**, **Devices**, or **Dialers** file, the default file is used. Any files specified in **Sysfiles** to serve as **Systems**, **Devices**, or **Dialers** files need to conform to the syntax used in the default files, `/etc/uucp/Systems`, `/etc/uucp/Devices` or `/etc/uucp/Dialers`.

Files

- **/etc/uucp** Contains all the default configuration files for BNU, including the **Sysfiles** file.
- **/etc/uucp/Sysfiles** Contains information about alternate **Systems**, **Devices** and **Dialers** files.
- **/etc/uucp/Systems** Lists and describes remote systems accessible to a local system, using the Basic Networking Utilities (BNU).
- **/etc/uucp/Devices** Contains information about available devices.
- **/etc/uucp/Dialers** Contains dialing sequences for various types of modems and other types of dialers.

Related Information

The **uucico** daemon, **ct** command, **cu** command, **uucp** command, **uux** command, **uusend** command

The **/etc/uucp/Devices** File, **/etc/uucp/Dialers** File, **/etc/uucp/Systems** File,

**Systems File Format for BNU**

**Purpose**

Lists and describes remote systems accessible to a local system, using the Basic Networking Utilities (BNU).
Description

BNU Systems files, /etc/uucp/Systems by default, list the remote computers with which users of a local system can communicate using the Basic Networking Utilities (BNU) program. Other files specified in the /etc/uucp/Sysfiles file can be configured and BNU Systems files. Each entry in a Systems file represents a remote system, and users on the local system cannot communicate with a remote system unless that system is listed in the local Systems file. A Systems file must be present on every computer at your site that uses the BNU facility.

Each entry in a Systems file contains:

- Name of the remote system
- Times when users can connect to the remote system
- Type of link (direct line or modem link)
- Speed of transmission over the link
- Information needed to log in to the remote system

Notes:
1. When a remote system not listed in a Systems file attempts to contact the remote system, the BNU program calls the /usr/sbin/uucp/remote.unknown shell procedure.
2. Only someone with root user authority can edit a Systems file, which is owned by the uucp program login ID.

Fields in a Systems File

Each entry in a Systems file is a logical line containing fields and optional subfields. These fields appear in the following order:

SystemName Time RetryTime Type ConversationProtocol Class Phone Login

There must be an entry in every field of a line in a Systems file. If a field does not apply to the particular remote system (for example, a hardwired connection would not need a telephone number in the Phone field), use a - (minus sign) as a placeholder.

Lines in a Systems file cannot wrap. In addition, each entry must be on only one line in the file. However, a Systems file can contain blank lines and comment lines. Comment lines begin with a # (pound sign). Blank lines are ignored.

System Name

The SystemName field contains the name of the remote system. You can list an individual remote system in a Systems file more than once. Each additional entry for a system represents an alternate communication path that the BNU program uses in sequential order when trying to establish a connection between the local and the remote system.

Time

The Time field contains a string that indicates the days of the week and the times of day during which users on the local system can communicate with the specified remote system. For example, the MoTuTh0800-1730 string indicates that local users can contact the specified remote system on Mondays, Tuesdays, and Thursdays from 8 a.m. until 5:30 p.m.

The day part of the entry can be a list including any day or days represented by Mo, Tu, We, Th, Fr, Sa, or Su. The day entry may also be Wk if users can contact the remote system on any weekday, or Any if they can use the remote system on any day of the week including Saturday and Sunday.
Enter the time at which users can contact the remote system as a range of times, using the 24-hour clock notation. For example, if users can communicate with the specified remote system only during the morning hours, type a range such as 0800-1200. If users can contact the remote computer at any time of day or night, simply leave the time range blank.

It is also possible to specify times during which users cannot communicate with the remote system by specifying a time range that spans 0000. For example, typing 0800-0600 means that users can contact the specified system at any time except between 6 a.m and 8 a.m. This is useful if a free line is needed at a certain time of day in order to use the remote system for administrative purposes.

If the remote system calls the local system, but users on the local system cannot call the remote system, the time entry may be Never.

Multiple Time fields are separated by a , (comma). For example, Wk1800-0600,Sa,Su means that users can contact the remote system on any weekday at any time except between the hours of 6 p.m. and 6 a.m. and at any time on Saturday and Sunday.

**RetryTime Subfield:** The RetryTime subfield is an optional subfield that specifies the minimum time in minutes between an unsuccessful attempt to reach the remote system and the retry time when the BNU program again attempts to communicate with that system. This subfield is separated from the rest of the string by a ; (semicolon). For example, Wk1800-0600,Sa,Su;2 indicates that if the first attempt to establish communications fails, BNU should continue to attempt to contact the remote system at no less than 2-minute intervals.

**Notes:**
1. This subfield, when present, overrides the default retry time of 5 minutes.
2. The retry time does not cause BNU to attempt contact with the system once the time has elapsed. It specifies the minimum time BNU must wait before attempting to contact the remote system.

**Type**
The **Type** field identifies the type of connection used to communicate with the remote system. The available types of connections are ACU for a telephone connection using a modem, the remote system name (as in the SystemName field) for a hardwired connection, and TCP for a connection using TCP/IP. There must be a corresponding entry for the type of connection in either the /etc/uucp/Devices file or the Devices file specified in the /etc/uucp/Sysfiles file.

**Conversation Protocol Subfield:** If you use the TCP entry in the Type field, the ConversationProtocol subfield, associated with the caller, specifies a conversation protocol. The default is the g protocol. To use a different subfield, enter a , (comma) and the letter representing one of the other conversation protocols, either t or e. These protocols are faster and more efficient than the g protocol.

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>g</td>
<td>This is the default. The g protocol is preferred for modem connections, but it involves a large overhead in running BNU commands because it uses the checksumming and packetizing functions.</td>
</tr>
<tr>
<td>t</td>
<td>The t protocol presumes an error-free channel and is essentially the g protocol without the checksumming and packetizing functions. Use the t protocol:</td>
</tr>
<tr>
<td></td>
<td>• To communicate with a site running the operating system version of the BNU program</td>
</tr>
<tr>
<td></td>
<td>• To communicate with a site running the Berkeley version of the UNIX-to-UNIX Copy Program (UUCP).</td>
</tr>
<tr>
<td></td>
<td>The t protocol cannot be used when the Type field is ACU or when a modem connection is being used.</td>
</tr>
</tbody>
</table>
**Protocol**  
**Explanation**

Use the `e` protocol:
- To communicate with a site running the BNU program on
- To communicate with a site running the operating system version of the BNU program.

The `e` protocol is not reliable for modem connections.

Use either the `t` or `e` protocol to communicate with a site running the operating system version of the BNU program. Use the `e` protocol for a site running a non-operating system version of the BNU program. Use the `t` protocol for sites running the Berkeley version of the UNIX-to-UNIX Copy Program (UUCP).

**Class**

The *Class* field typically specifies the speed at which the specified hardwired or telephone line transmits data. It is generally 300, 1200, 2400, or higher for a hardwired device, and 300, 1200, or 2400 for a telephone connection.

This field can also contain a letter with a speed (for example, `C1200`, `D1200`) to differentiate between classes of dialers. For example, some offices have more than one telephone network, one for internal use and one for external communications. In such a case, it is necessary to distinguish which lines should be used for each connection.

If the entry in the *Type* field is `ACU`, the *Class* field in a *Systems* file is matched against the *Class* field in a *Devices* file to find the device to use for connections. For example, if a *Systems* file entry for system `hera` is:

```
hera  Any  ACU   1200  3-3-5-2  login: nuucp  sssword: oldoaktree
```

BNU searches for an entry in the *Devices* file with a *Type* of `ACU` and a *Class* of 1200 and connects to system `hera` using the first available device that meets these specifications.

If the device can match any speed, enter the word *Any* in the *Class* field. Note that the word *Any* begins with an uppercase `A`.

Do not include a transmission rate for a TCP/IP connection. If you do not type a transmission rate in the *Class* field, use a `-` (minus sign) as a placeholder.

**Phone**

For a telephone connection over a modem, the *Phone* field specifies the telephone number used to reach the remote modem. If this entry represents a hardwired connection, type a `-` (minus sign) as a placeholder. If this entry represents a telephone connection using a modem, type the remote modem’s phone number.

The *Phone* field for a telephone connection must include all of the following items that apply, in the following order:

1. Outside line code
2. Long-distance access codes
3. Number 1 (one) plus the area code (if the modem is out of the local area)
4. Three-digit exchange number
5. Four-digit modem number

Entering a complete phone number is the most efficient method of including phone numbers if your site uses only a relatively small number of telephone connections. However, if your site includes a large number of remote connections established using a phone line and a modem, you may prefer to use the `/etc/uucp/Dialcodes` file to set up dial-code abbreviations.
For example, if your site communicates regularly using modems to other systems at the same remote site, it is more efficient to use a dial-code abbreviation in a Systems file than to type the complete phone number of each remote modem.

The dial-code entry in the /etc/uucp/Dialcodes file defines an alphabetic abbreviation that represents the following portions of the phone number:

- Outside line code
- Long-distance access code
- Number 1 (one) plus the area code (if the modem is out of the local area)
- Three-digit exchange number

In the Phone field in a Systems file entry, type the alphabetic abbreviation followed by the four-digit modem number.

**Note:** Enter the alphabetic abbreviation in the /etc/uucp/Dialcodes file only once for all the remote modems listed in a Systems file. Then use the same abbreviation for all entries in a Systems file for modems at that site.

For callers that are actually switches, the Phone field is the token the switch requires to get to the particular computer. The token you enter here is used by the functions specified in the Type field of the /etc/uucp/Dialcodes file.

**Login**

The Login field specifies login information that the remote system must receive before allowing the calling local system to establish a connection. The Login field is a series of fields and subfields called expect-send characters.

**Expect-Send Characters in Login Fields:** Enter the required login information as:

```
[Expect Send] ...
```

The Expect subfield contains characters that the local system expects to receive from the remote system. Once the local system receives those characters, it sends another string of characters that comprise the Send subfield.

For example, the first Expect subfield generally contains the remote system’s login prompt, and the first Send subfield generally contains the remote system login ID. The second Expect subfield contains the remote password prompt, and the second Send subfield contains the remote system password.

The Expect subfield may include subfields entered in the following form:

```
Expect[-Send-Expect] ...
```

In this case, the first Expect subfield still represents the string that the local system expects to receive from the remote system. However, if the local system does not receive (or cannot read) the first Expect string, it sends its own string (the Send string within brackets) to the remote system. The local system then expects to receive another Expect string from the remote system.

For example, the Expect string may contain the following characters:

```
login:--login:
```

The local system expects to receive the login: string. If the remote system sends that string and the local system receives it correctly, the BNU program goes on to the next field in the expect-send sequence. However, if the local system does not receive the login: string, it sends a null character followed by a new line, and then expects to receive a second login: string from the remote computer.
If the remote system does not send an Expect string to the local system, type "" (two double quotation marks), representing a null string, in the first Expect subfield.

Every time the local system sends a field, it automatically transmits a new line following that Send subfield. To disable this automatic new line, type \c (backslash and the letter c) as the last two characters in the Send string.

Two special strings can be included in the login sequence. The EOT string sends an ASCII EOT (end of transmission) character, and the BREAK string attempts to send an ASCII BREAK character.

**Valid Expect-Send Sequences:** Following are the valid expect-send strings for the Login field:

<table>
<thead>
<tr>
<th>String</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>\N</td>
<td>Null character.</td>
</tr>
<tr>
<td>\b</td>
<td>Backspace character.</td>
</tr>
<tr>
<td>\c</td>
<td>At the end of a field, suppress the new line that normally follows the characters in a Send subfield. Otherwise, ignore this string.</td>
</tr>
<tr>
<td>\d</td>
<td>Delay 2 seconds before sending or reading more characters.</td>
</tr>
<tr>
<td>\p</td>
<td>Pause for approximately .25 to .50 seconds.</td>
</tr>
<tr>
<td>\E</td>
<td>Turn on the echo check.</td>
</tr>
<tr>
<td>\e</td>
<td>Turn off the echo check.</td>
</tr>
<tr>
<td>\K</td>
<td>Send a BREAK character. This is the same as entering BREAK. This character can be used to cycle a modem’s speed.</td>
</tr>
<tr>
<td>\n</td>
<td>New-line character.</td>
</tr>
<tr>
<td>\r</td>
<td>Carriage return.</td>
</tr>
<tr>
<td>\s</td>
<td>Space character.</td>
</tr>
<tr>
<td>\t</td>
<td>Tab character.</td>
</tr>
<tr>
<td>\</td>
<td>Backslash character.</td>
</tr>
<tr>
<td>EOT</td>
<td>EOT character. When you enter this string, the system sends two EOT new-line characters.</td>
</tr>
<tr>
<td>BREAK</td>
<td>BREAK character. This character can be used to cycle the modem speed.</td>
</tr>
<tr>
<td>\ddd</td>
<td>Collapse the octal digits (\ddd) into a single character and send that character.</td>
</tr>
</tbody>
</table>

**Using the BREAK Character to Cycle a Modem:** A BREAK or \K character is usually sent to cycle the line speed on computers that have a multispeed modem. For example, if you are using a 2400 baud modem to contact a remote system with a multi speed modem that normally answers the phone at 9600 baud, you can begin the chat script for that system with a \K character to cause the remote system modem to cycle down to 2400 baud.

**Entries for Use with TCP/IP**

If your site is using TCP/IP, include the relevant TCP/IP entries in a Systems file. For a remote system connected to the local system using TCP/IP, the entries in the SystemName, Time, and Login fields are the same as for a remote system using any other type of connection. For the Type field, decide on the appropriate TCP/IP conversation protocol to enter in the TCP ConversationProtocol subfield. Enter TCP followed by a ,, (comma) followed by the letter representing the protocol. In the Class and Phone fields, enter a - (minus sign) as a placeholder.

**Examples**

**Setting Up Entries Using Modems**

1. A standard entry for a telephone connection using a modem looks like this:
   ```
   merlin 0830-1730 ACU 1200 123-4567 in:--in: uucp1 word: rainday
   ```
   This entry allows users to contact system merlin daily between 8:30 a.m. and 5:30 p.m., using an ACU at 1200 bps. The telephone number is 123-4567. The login name on merlin is uucp1 and the password
is rain day. The local system expects the phrase in: before it sends the login name. If the local system does not receive the phrase in:, it sends a null character and a new-line character and expects the phrase again.

2. To use a 1200 baud modem to contact a system with a multispeed modem, make an entry similar to the following:
   athena Any ACU 1200 123-7654 \K\K in:--in: uucpa word: shield
   The \K prefac ing the login script instructs the remote modem to cycle down one speed. If the modem has three speeds, 9600, 2400, and 1200, the first \K character causes it to cycle to the 2400 baud setting, and the second \K character causes it to use the 1200 baud setting. (A third \K causes the modem to start the cycle over by returning to 9600 baud.)

**Setting Up Entries Using Direct Connections**

A standard entry for a hardwired connection between a local and a remote system looks like this:
   hera Any hera 1200 - login:--login: uzeus word: thunder

The remote system is hera, which can be called at any time. The entry in the Type field is also hera, indicating a directory connection at 1200 bps (the Class field). There is a placeholder in the Phone field since no telephone number is necessary.

**Setting Up Entries Using TCP/IP Connections**

In order to make the appropriate entries in a Systems file, decide on the appropriate TCP/IP conversation protocol to enter in the TCP Caller subfield. For example, enter the following in a Systems file to use TCP/IP to connect to system venus with the default g protocol:
   venus Any TCP - - in:--in: uzeus word: lamplight

Replace the *send* and *expect* characters in the sample login field with the login prompt, login, password prompt, and password appropriate to the remote system for which you are establishing a connection.

**Using Dialcode Abbreviations**

To use a dialcode abbreviation defined in the /etc/uucp/Dialcodes file, enter the following in a Systems file:
   merlin Any ACU 1200 local8784 in:--in: uucpl word: magic

This assumes that an entry for the dial code local exists in the Dialcodes file. For example, the following entry:
   local 9=445

in the Dialcodes file would cause BNU to expand the telephone number as 9=4458784.

**Setting Up Entries for Both Local and Remote Systems**

For a direct connection between two systems, a Systems file on system zeus contains the following entry for the remote system hera:
   hera Any hera 1200 - "\r\d\r\d\r in:--in: uzeus word: thunder

A Systems file on system hera contains the following entry for system zeus:
   zeus Any zeus 1200 - "\r\d\r\d\r in:--in: uhera word: lostleaf

**Files**

/etc/uucp directory

Contains all the configuration files for BNU, including a Systems file.

Specifies possible alternative files for the /etc/uucp/Systems file.

Contains information about available devices.
/etc/uucp/Dialcodes file
/etc/uucp/Permissions file
/usr/sbin/uucp/remote.unknown file

Contains dialing code abbreviations.
Describes access permissions for remote systems.
Records contacts from unknown systems.

Related Information
The mail command, sendmail command, uucp command, uucpadm command, uname command, uuto command, uutry command, Uutry command, uukick command, uux command.

The uucico daemon, uucpd daemon, uusched daemon, uuxqt daemon.

Configuring BNU | Monitoring a BNU remote connection | Debugging BNU login failures using the uucico daemon | Understanding the BNU File and Directory Structure in Networks and communication management.

---

telnet.conf File Format for TCP/IP

Purpose
Translates a client's terminal-type strings into terminfo file entries.

Description
The telnetd daemon uses the /etc/telnet.conf file during terminal negotiation to translate a client's terminal-type strings into terminfo file entries. The telnet.conf file is used when a client's terminal does not correspond directly to a terminfo file entry. If this is the case, the telnet.conf file can map standard terminal names (defined in RFC-1060 Assigned Numbers) to terminfo file entries that the system can emulate.

Each line in the telnet.conf file can contain up to 255 characters. Lines beginning with a # (pound sign) are comment lines.

The telnet.conf file is structured in a two-column line format, with dashes separating the items in each column. The first column specifies a manufacturer, model type, and optional additional information. The second column specifies the terminfo file entry that corresponds to the manufacturer, model, and optional information in the first column. The items in the first column can be either uppercase or lowercase. The items in the second column must be lowercase. RFC-1060 specifies the first terminal type in the telnet.conf file. The format for the telnet.conf file is:

Manufacturer-Model-Options TerminfoModel-Options

Security
Suggested permissions for the telnet.conf file are rw-rw-- or 664. Suggested ownership is root for owner and system for group.

Examples
Sample telnet.conf entries might look like the following:
DEC-VT100-AM vt100-am
diablo-1620-m8 1620-m8
h-19-a 19-a
TI-800 ti-800

In the first entry, the manufacturer is DEC (Digital Equipment Corporation), the model is VT100, and the AM option specifies automargin. In the second entry, the manufacturer is diablo, the model is 1620, and the m8 option specifies a left margin of 8 columns. In the third entry, the manufacturer is h (Heath), the model is

586 AIX 5L Version 5.3 Files Reference
19, and the `a` option specifies ANSI mode. In the fourth entry, the manufacturer is TI (Texas Instruments), and the model is 800; no options are specified. For additional `terminfo` options, refer to the `*.ti` files in the `/usr/lib/terminfo` directory.

**Files**

`terminfo` Describes terminal by capability.

**Related Information**

The `telnet` command.

The `telnetd` daemon.

**terminfo Directory**

**Purpose**

Contains compiled `terminfo` source files.

**Description**

`terminfo` is a compiled database describing the capabilities of terminals. Terminals are described in the `terminfo` source files via entries. Each entry contains information about the capabilities for a particular terminal or set of common terminals. Capabilities include the operations that can be performed, the padding requirements, cursor positioning, command sequences, and initialization sequences.

The compiled `terminfo` database is used by applications such as curses and `vi` that must have knowledge of the terminal but do not want to be terminal-dependent.

An example of a `terminfo` source file is provided.

This article explains the `terminfo` source file format. Before a `terminfo` description can be used by applications, the `terminfo` source file it resides in must be compiled using the `tic` command. Using the `tic` command results in the creation of one or more binaries, one for each terminal. The collection of `terminfo` binaries in a directory (usually `/usr/share/lib/terminfo`) is known as the `terminfo` database, or `terminfo`.

**Source File Entries**

You can edit or modify source files. A source file can contain one or more terminal descriptions or entries. A `terminfo` source file has a `.ti` suffix. Examples of source files are the `/usr/share/lib/terminfo/ibm.ti` file, which describes IBM terminals, and the `/usr/share/lib/terminfo/dec.ti` file, which describes DEC terminals.

See the `infocmp` command for obtaining the source description for a terminal when only the binary is available.

Each entry in a `terminfo` source file consists of a number of fields separated by commas. White space between commas is ignored. The following example shows a source file entry:

```
ibm6155-113|IBM 6155 Black & White display,
  font0=%[10m,  font1=%[11m,  font2=%[12m,
  bold=%[12m,  sgr0=%[0;10m,  cols#113,
  lines#38,
  sgr=%[7;%?%2t;4;%?%3t;7;%?%4t;5;%?%6t;12%m,
  blink0,  use=ibm5151,
```
Entries can continue onto multiple lines by placing white space at the beginning of each subsequent line. To create a comment line, begin the line with a # (pound sign) character. To comment out an individual terminal capability, put a period before the capability name.

The first field (or line) for each terminal gives the various names by which the terminal is known, separated by | (pipe symbol) characters. The first name given should be the most common abbreviation for the terminal. (This name is the one most commonly used when setting the TERM environment variable.) The last name given should be a long name fully identifying the terminal. All other names are understood as synonyms for the terminal name. All names but the last should contain no blanks. The last name may contain blanks for readability. All names should be unique.

The remaining fields identify the terminal’s capabilities.

When choosing terminal names, there are some conventions you should follow. The root name should represent the particular hardware class of the terminal. Do not use hyphens in the root name, except to avoid synonyms that conflict with other names. To indicate possible modes for the hardware or user preferences, append a - (minus sign) and one of the following suffixes:

<table>
<thead>
<tr>
<th>Suffix</th>
<th>Meaning</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>-am</td>
<td>With automatic margins (usually default)</td>
<td>Terminal-am</td>
</tr>
<tr>
<td>-m</td>
<td>Monochrome mode</td>
<td>Terminal-m</td>
</tr>
<tr>
<td>-w</td>
<td>Wide mode (more than 80 columns)</td>
<td>Terminal-w</td>
</tr>
<tr>
<td>-nam</td>
<td>Without automatic margins</td>
<td>Terminal-nam</td>
</tr>
<tr>
<td>-n</td>
<td>Number of lines on the screen</td>
<td>Terminal-60</td>
</tr>
<tr>
<td>-na</td>
<td>No arrow keys (leave them in local)</td>
<td>Terminal-na</td>
</tr>
<tr>
<td>-rp</td>
<td>Number of pages of memory</td>
<td>Terminal-4p</td>
</tr>
<tr>
<td>-rv</td>
<td>Reverse video</td>
<td>Terminal-rv</td>
</tr>
<tr>
<td>-s</td>
<td>Status line simulation. The terminal allows for one or more lines that are normally part of the screen to be used for the status line. This is not the same as terminals that have permanently dedicated status lines.</td>
<td>Terminal-s</td>
</tr>
<tr>
<td>-unk</td>
<td>Unknown mode. This entry can be used to define a general description of a terminal that has several of the modes described above. The other entries would use the unknown entry as a base description and add the appropriate customization. See the use= field.</td>
<td>Terminal-unk</td>
</tr>
</tbody>
</table>

A terminal in 132-column mode would be Terminal-w.

**Types of Capabilities**

A `terminfo` entry can define any number of capabilities. All capabilities belong to one of three types:

- **Boolean**: Indicates that the terminal has a particular feature. Boolean capabilities are true if the corresponding name is contained in the terminal description.
- **Numeric**: Gives the size of the terminal or the size of particular delays.
- **String**: Gives a sequence that can be used to perform particular terminal operations.
This article provides tables that document the capability types. All the tables list the following:

- **Variable**: The name the application uses to access a capability.
- **Cap Name**: The short capability name. This name is used in the `terminfo` database text and by the person creating or editing a source file entry. You can use the `tput` command to output the value of a capability for a particular terminal.
- **I.Code**: The 2-letter internal code used in the compiled database. This code always corresponds to a `termcap` capability name.
- **Description**: A description of the capability.

Capability names have no absolute length limit. An informal limit of five characters is adopted to keep them short and to allow the tabs in the caps source file to be aligned. Whenever possible, names are chosen to be the same as or similar to the ANSI X3.64 standard of 1979.

### Boolean Capabilities

A Boolean capability indicates that the terminal has some particular feature. For instance, the `am` capability in a terminal description indicates that the terminal has automatic margins (such as an automatic new line when the end of a line is reached). The following are the Boolean capabilities:

**Table 8. Boolean Capabilities**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Cap Name</th>
<th>I.Code</th>
</tr>
</thead>
</table>
| auto_left_margin               | {
\begin{itemize}
  \item auto_left_margin indicates \texttt{cub1} wraps from column 0 to last column.
\end{itemize} |
|                               | bw                | bw     |
| auto_right_margin              | {
\begin{itemize}
  \item auto_right_margin indicates terminal has automatic margins.
\end{itemize} |
|                               | am                | am     |
| back_color_erase               | {
\begin{itemize}
  \item back_color_erase erases screen with current background.
\end{itemize} |
|                               | bce               | ut     |
| can_change                      | {
\begin{itemize}
  \item can_change can redefine existing color.
\end{itemize} |
|                               | ccc               | cc     |
| ceol_standout_glitch            | {
\begin{itemize}
  \item ceol_standout_glitch indicates that standout is not erased by overwriting.
\end{itemize} |
|                               | xhp               | xs     |
| col_addr_glitch                 | {
\begin{itemize}
  \item col_addr_glitch indicates only positive motion for hpa/mhpa caps.
\end{itemize} |
|                               | xhp               | YA     |
| cpi_changes_res                 | {
\begin{itemize}
  \item cpi_changes_res indicates resolution changed when changing character pitch.
\end{itemize} |
|                               | cpix              | YF     |
| cr_cancels_micro_mode           | {
\begin{itemize}
  \item cr_cancels_micro_mode indicates cr turns off micro mode.
\end{itemize} |
|                               | crxm              | YB     |
| dest_tabs_magic_smso (or teleray_glitch) | {
\begin{itemize}
  \item dest_tabs_magic_smso (or teleray_glitch) indicates destructive tabs and blanks inserted while entering standout mode.
\end{itemize} |
|                               | xt                | xt     |
| eat_newline_glitch              | {
\begin{itemize}
  \item eat_newline_glitch ignores new-line character after 80 columns.
\end{itemize} |
|                               | xenl              | xn     |
| erase_overstrike                | {
\begin{itemize}
  \item erase_overstrike erases overstrikes with a blank.
\end{itemize} |
|                               | eo                | eo     |
| generic_type                    | {
\begin{itemize}
  \item generic_type indicates generic line type, such as dialup or switch.
\end{itemize} |
<p>|                               | gn                | gn     |</p>
<table>
<thead>
<tr>
<th>Variable</th>
<th>Cap Name</th>
<th>I.Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hard_copy</td>
<td>hc</td>
<td>hc</td>
<td>Indicates hardcopy terminal.</td>
</tr>
<tr>
<td>hard_cursor</td>
<td>chts</td>
<td>HC</td>
<td>Indicates cursor is hard to see.</td>
</tr>
<tr>
<td>has_meta_key</td>
<td>km</td>
<td>km</td>
<td>Indicates terminal has a meta key, such as shift or sets parity bit.</td>
</tr>
<tr>
<td>has_print_wheel</td>
<td>daisy</td>
<td>YC</td>
<td>Indicates operator needed to change character set.</td>
</tr>
<tr>
<td>has_status_line</td>
<td>hs</td>
<td>hs</td>
<td>Indicates terminal has a dedicated status line.</td>
</tr>
<tr>
<td>hue_lightness_saturation</td>
<td>hls</td>
<td>hl</td>
<td>Uses HLS color notation (Tektronix).</td>
</tr>
<tr>
<td>insert_null_glitch</td>
<td>in</td>
<td>in</td>
<td>Indicates insert mode distinguishes nulls.</td>
</tr>
<tr>
<td>lpi_changes_res</td>
<td>lpix</td>
<td>YG</td>
<td>Indicates resolution changed when changing line pitch.</td>
</tr>
<tr>
<td>memory_above</td>
<td>da</td>
<td>da</td>
<td>Display retained above the screen (usually multi-page terminals).</td>
</tr>
<tr>
<td>memory_below</td>
<td>db</td>
<td>db</td>
<td>Display retained below the screen (usually multi-page terminals)</td>
</tr>
<tr>
<td>move_insert_mode</td>
<td>mir</td>
<td>mi</td>
<td>Indicates safe to move while in insert mode.</td>
</tr>
<tr>
<td>move_standout_mode</td>
<td>msgr</td>
<td>ms</td>
<td>Indicates safe to move in standout modes.</td>
</tr>
<tr>
<td>needs_xon_xoff</td>
<td>nxon</td>
<td>nx</td>
<td>Indicates padding will not work, that xon/xoff is required.</td>
</tr>
<tr>
<td>no_esc_ctlc (or beehive_glitch)</td>
<td>xsb</td>
<td>xb</td>
<td>Indicates a terminal with F1=escape and F2=Ctrl-C.</td>
</tr>
<tr>
<td>no_pad_char</td>
<td>npc</td>
<td>NP</td>
<td>Indicates pad character does not exist.</td>
</tr>
<tr>
<td>non_dest_scroll_region</td>
<td>ndscr</td>
<td>ND</td>
<td>Indicates non-destructive scrolling region.</td>
</tr>
<tr>
<td>non_rev_rmcup</td>
<td>nrrmc</td>
<td>NR</td>
<td>Indicates smcup does not reverse rmcup.</td>
</tr>
<tr>
<td>over_strike</td>
<td>os</td>
<td>os</td>
<td>Indicates terminal overstrikes.</td>
</tr>
<tr>
<td>Capability</td>
<td>Code</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------------------------</td>
<td>------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>prtr_silent</td>
<td>mc5i</td>
<td>Indicates printer will not echo on screen.</td>
<td></td>
</tr>
<tr>
<td>row_addr_glitch</td>
<td>xvpa</td>
<td>Indicates only positive motion for vpa/mvpa caps.</td>
<td></td>
</tr>
<tr>
<td>semi_auto_right_margin</td>
<td>sam</td>
<td>Indicates printing in last column causes carriage return.</td>
<td></td>
</tr>
<tr>
<td>status_line_esc_ok</td>
<td>eslok</td>
<td>Indicates escape can be used on the status line.</td>
<td></td>
</tr>
<tr>
<td>tilde_glitch</td>
<td>hz</td>
<td>Indicates terminal cannot print the ~ (tilde) character.</td>
<td></td>
</tr>
<tr>
<td>transparent_underline</td>
<td>ul</td>
<td>Overstrikes with underline character.</td>
<td></td>
</tr>
<tr>
<td>xon_xoff</td>
<td>xon</td>
<td>Indicates terminal uses xon/xoff handshaking.</td>
<td></td>
</tr>
</tbody>
</table>

### Numeric Capabilities

Numeric capabilities are followed by the # (pound sign) character and a numeric value. The **cols#80** capability indicates the terminal has 80 columns. The following are the numeric capabilities:

**Table 9. Numeric Capabilities**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Cap Name</th>
<th>I.Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>buffer_capacity</td>
<td>bufsz</td>
<td>Ya</td>
</tr>
<tr>
<td>columns</td>
<td>cols</td>
<td>co</td>
</tr>
<tr>
<td>dot_horz_spacing</td>
<td>spinh</td>
<td>Yc</td>
</tr>
<tr>
<td>dot_vert_spacing</td>
<td>spinv</td>
<td>Yb</td>
</tr>
<tr>
<td>init_tabs</td>
<td>it</td>
<td>it</td>
</tr>
<tr>
<td>label_height</td>
<td>lh</td>
<td>lh</td>
</tr>
<tr>
<td>label_width</td>
<td>lw</td>
<td>lw</td>
</tr>
<tr>
<td>lines</td>
<td>lines</td>
<td>li</td>
</tr>
<tr>
<td>lines_of_memory</td>
<td>lm</td>
<td>lm</td>
</tr>
<tr>
<td>magic_cookie_glitch</td>
<td>xmc</td>
<td>sg</td>
</tr>
</tbody>
</table>
### Table 9. Numeric Capabilities (continued)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Cap Name</th>
<th>I.Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>max_attributes</td>
<td>ma</td>
<td>ma</td>
</tr>
<tr>
<td>Identifies the maximum combined video attributes the terminal can display.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>max_colors</td>
<td>colors</td>
<td>Co</td>
</tr>
<tr>
<td>Specifies the maximum number of colors supported.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>max_micro_address</td>
<td>maddr</td>
<td>Yd</td>
</tr>
<tr>
<td>Indicate the limit on use of mhp and mvpa.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>max_micro_jump</td>
<td>mjump</td>
<td>Ye</td>
</tr>
<tr>
<td>Specifies the limit on use of the mcub1, mcuf1, mcuu1, and mcud1 capabilities.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>max_pairs</td>
<td>pairs</td>
<td>pa</td>
</tr>
<tr>
<td>Specifies the maximum number of color pairs supported.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>maximum_windows</td>
<td>wnum</td>
<td>MW</td>
</tr>
<tr>
<td>Specifies the maximum number of defineable windows.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>micro_char_size</td>
<td>mcs</td>
<td>Yf</td>
</tr>
<tr>
<td>Specifies the character step size when in micro mode.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>micro_line_size</td>
<td>mls</td>
<td>Yg</td>
</tr>
<tr>
<td>Identifies the line step size when in micro mode.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>no_color_video</td>
<td>ncv</td>
<td>NC</td>
</tr>
<tr>
<td>Indicates video attributes that cannot be used with colors.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>num_labels</td>
<td>nlab</td>
<td>NI</td>
</tr>
<tr>
<td>Specifies the number of labels on the screen. This value starts at 1.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>number_of_pins</td>
<td>npins</td>
<td>Yh</td>
</tr>
<tr>
<td>Identifies the number of pins in the print-head.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>output_res_char</td>
<td>orc</td>
<td>Yi</td>
</tr>
<tr>
<td>Specifies the horizontal resolution in units per character.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>output_res_horz_inch</td>
<td>orhi</td>
<td>Yk</td>
</tr>
<tr>
<td>Specifies the horizontal resolution in units per inch.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>output_res_line</td>
<td>orl</td>
<td>Yj</td>
</tr>
<tr>
<td>Specifies the vertical resolution in units per line.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>output_res_vert_inch</td>
<td>orvi</td>
<td>Yl</td>
</tr>
<tr>
<td>Indicates vertical resolution in units per inch.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>padding_baud_rate</td>
<td>pb</td>
<td>pb</td>
</tr>
<tr>
<td>Indicates lowest baud rate where carriage-return and line-return padding is needed.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>print_rate</td>
<td>cps</td>
<td>Ym</td>
</tr>
<tr>
<td>Indicates print rate in characters per second.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>virtual terminal</td>
<td>vt</td>
<td>vt</td>
</tr>
<tr>
<td>--------------------------</td>
<td>---------</td>
<td>---------</td>
</tr>
<tr>
<td>Indicates virtual terminal number.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>wide_char_size</td>
<td>widcs</td>
<td>Yn</td>
</tr>
<tr>
<td>Identifies the character step size when the terminal is in double-wide mode.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>width_status_lines</td>
<td>wsl</td>
<td>ws</td>
</tr>
<tr>
<td>Specifies the number of columns in status lines.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**String Capabilities**

You define string-valued capabilities, such as the `el` capability (clear to end of line) with a 2-character code, an `=` (equal sign), and a string ending with a `,` (comma). A delay in milliseconds can appear anywhere in a string capability. To define a delay, enclose the delay between a `$<` and a `>`. The following shows the `el` capability with a delay of 3:

```
etl=\EK$<3>
```

The `puts` subroutine provides padding characters for a delay. A delay can be a number, such as 20, or a number followed by an `*` (asterisk), such as 3*. An asterisk indicates that the required padding is proportional to the number of lines affected by the operation. The number given represents the required padding for each affected unit. (For insert character, the factor is the number of lines affected, which is always 1, unless the terminal has the `xenl` capability and the software supports it). If you specify an asterisk, it is sometimes useful to give a delay of the form `a,b`, such as 3.5, to specify a delay for each unit to tenths of milliseconds. You can only specify one decimal place.

The `terminfo` database provides several escape sequences in the string-valued capabilities for easy encoding of characters. The following escape codes are recognized:

<table>
<thead>
<tr>
<th>Escape Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>\E,\e</td>
<td>Escape</td>
</tr>
<tr>
<td>\n</td>
<td>New line</td>
</tr>
<tr>
<td>\r</td>
<td>Line feed</td>
</tr>
<tr>
<td>\t</td>
<td>Tab</td>
</tr>
<tr>
<td>\b</td>
<td>Backspace</td>
</tr>
<tr>
<td>\f</td>
<td>Form feed</td>
</tr>
<tr>
<td>\s</td>
<td>Space</td>
</tr>
<tr>
<td>^</td>
<td>Caret</td>
</tr>
<tr>
<td>\ backslash</td>
<td>Backslash</td>
</tr>
<tr>
<td>,</td>
<td>Comma</td>
</tr>
<tr>
<td>:</td>
<td>Colon</td>
</tr>
<tr>
<td>\nnn</td>
<td>Character with octal value nnn</td>
</tr>
<tr>
<td>^x</td>
<td>Ctrl-x for any appropriate x</td>
</tr>
<tr>
<td>\0</td>
<td>Null character. \0 actually produces \200, which does not end a string but behaves as a null character on most terminals.</td>
</tr>
</tbody>
</table>

The following conventions are used in the String Capabilities table:

(G) Indicates that the string is passed through `tparm`, with parameters as given (#i).

(*) Indicates that padding can be based on the number of lines affected.

(#i) Indicates the ith parameter.
## Table 10. String Capabilities

<table>
<thead>
<tr>
<th>Variable</th>
<th>Cap Name</th>
<th>I.Code</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>appl_defined_str</code></td>
<td><code>apstr</code></td>
<td><code>za</code></td>
</tr>
<tr>
<td>Application-defined terminal string.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>asc_chars</code></td>
<td><code>acsc</code></td>
<td><code>ac</code></td>
</tr>
<tr>
<td>Alternate character set mapping of glyph to characters.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>back_tab</code></td>
<td><code>cbt</code></td>
<td><code>bt</code></td>
</tr>
<tr>
<td>Back tab.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>bell</code></td>
<td><code>bel</code></td>
<td><code>bl</code></td>
</tr>
<tr>
<td>Produces an audible signal (bell).</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>box_chars_1</code></td>
<td><code>box1</code></td>
<td><code>bx</code></td>
</tr>
<tr>
<td>Box characters, primary set.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>box_chars_2</code></td>
<td><code>box2</code></td>
<td><code>by</code></td>
</tr>
<tr>
<td>Box characters, alternate set.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>box_attr_1</code></td>
<td><code>batt1</code></td>
<td><code>Bx</code></td>
</tr>
<tr>
<td>Attributes for <code>box_chars_1</code>.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>box_attr_2</code></td>
<td><code>batt2</code></td>
<td><code>By</code></td>
</tr>
<tr>
<td>Attributes for <code>box_chars_2</code>.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>carriage_return</code></td>
<td><code>cr</code></td>
<td><code>cr</code></td>
</tr>
<tr>
<td>Indicates carriage return. (*)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>change_char_pitch</code></td>
<td><code>cpi</code></td>
<td><code>ZA</code></td>
</tr>
<tr>
<td>Change number of characters per inch.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>change_line_pitch</code></td>
<td><code>lpi</code></td>
<td><code>ZB</code></td>
</tr>
<tr>
<td>Change number of lines per inch.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>change_res_horz</code></td>
<td><code>chr</code></td>
<td><code>ZC</code></td>
</tr>
<tr>
<td>Change horizontal resolution.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>change_res_vert</code></td>
<td><code>cvr</code></td>
<td><code>XD</code></td>
</tr>
<tr>
<td>Change vertical resolution.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>char_padding</code></td>
<td><code>rmp</code></td>
<td><code>rP</code></td>
</tr>
<tr>
<td>Specifies character padding when in replace mode.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>change_scroll_region</code></td>
<td><code>csr</code></td>
<td><code>cs</code></td>
</tr>
<tr>
<td>Changes scroll region to lines #1 through #2. (G)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>char_set_names</code></td>
<td><code>csnm</code></td>
<td><code>Zy</code></td>
</tr>
<tr>
<td>List of character set names.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>clear_all_tabs</code></td>
<td><code>tbc</code></td>
<td><code>ct</code></td>
</tr>
<tr>
<td>Clears all tab stops.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>clear_margins</code></td>
<td><code>mgc</code></td>
<td><code>MC</code></td>
</tr>
<tr>
<td>Clear left and right soft margins.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable</td>
<td>Cap Name</td>
<td>I.Code</td>
</tr>
<tr>
<td>---------------</td>
<td>----------</td>
<td>--------</td>
</tr>
<tr>
<td>clear_screen</td>
<td>clear</td>
<td>cl</td>
</tr>
<tr>
<td>clr_bol</td>
<td>el1</td>
<td>cb</td>
</tr>
<tr>
<td>clr_eol</td>
<td>el</td>
<td>ce</td>
</tr>
<tr>
<td>clr_eod</td>
<td>ed</td>
<td>cd</td>
</tr>
<tr>
<td>color_bg_0</td>
<td>colb0</td>
<td>d0</td>
</tr>
<tr>
<td>color_bg_1</td>
<td>colb1</td>
<td>d1</td>
</tr>
<tr>
<td>color_bg_2</td>
<td>colb2</td>
<td>d2</td>
</tr>
<tr>
<td>color_bg_3</td>
<td>colb3</td>
<td>d3</td>
</tr>
<tr>
<td>color_bg_4</td>
<td>colb4</td>
<td>d4</td>
</tr>
<tr>
<td>color_bg_5</td>
<td>colb5</td>
<td>d5</td>
</tr>
<tr>
<td>color_bg_6</td>
<td>colb6</td>
<td>d6</td>
</tr>
<tr>
<td>color_bg_7</td>
<td>colb7</td>
<td>d7</td>
</tr>
<tr>
<td>color_fg_0</td>
<td>colf0</td>
<td>c0</td>
</tr>
<tr>
<td>color_fg_1</td>
<td>colf1</td>
<td>c1</td>
</tr>
<tr>
<td>color_fg_2</td>
<td>colf2</td>
<td>c2</td>
</tr>
<tr>
<td>color_fg_3</td>
<td>colf3</td>
<td>c3</td>
</tr>
<tr>
<td>color_fg_4</td>
<td>colf4</td>
<td>c4</td>
</tr>
<tr>
<td>color_fg_5</td>
<td>colf5</td>
<td>c5</td>
</tr>
<tr>
<td>color_fg_6</td>
<td>colf6</td>
<td>c6</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-----------------------</td>
<td>------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><strong>color_fg_7</strong></td>
<td>Foreground color 7, white.</td>
<td></td>
</tr>
<tr>
<td><strong>column_address</strong></td>
<td>Sets cursor column. (G)</td>
<td></td>
</tr>
<tr>
<td><strong>command_character</strong></td>
<td>Indicates that a terminal command-prototype character can be set.</td>
<td></td>
</tr>
<tr>
<td><strong>create_window</strong></td>
<td>Define win #1 to go from #2, #3 to #4, #5.</td>
<td></td>
</tr>
<tr>
<td><strong>cursor_address</strong></td>
<td>Indicates screen-relative cursor motion row #1, col #2. (G)</td>
<td></td>
</tr>
<tr>
<td><strong>cursor_down</strong></td>
<td>Moves cursor down one line.</td>
<td></td>
</tr>
<tr>
<td><strong>cursor_home</strong></td>
<td>Moves cursor to home position (if no cup addressing).</td>
<td></td>
</tr>
<tr>
<td><strong>cursor_invisible</strong></td>
<td>Makes cursor invisible.</td>
<td></td>
</tr>
<tr>
<td><strong>cursor_left</strong></td>
<td>Moves cursor left one space.</td>
<td></td>
</tr>
<tr>
<td><strong>cursor_mem_address</strong></td>
<td>Indicates memory relative cursor addressing. (G)</td>
<td></td>
</tr>
<tr>
<td><strong>cursor_normal</strong></td>
<td>Makes cursor appear normal (undo vs or vi).</td>
<td></td>
</tr>
<tr>
<td><strong>cursor_right</strong></td>
<td>Indicates nondestructive space (cursor right).</td>
<td></td>
</tr>
<tr>
<td><strong>cursor_to_ll</strong></td>
<td>Moves cursor to first column of last line (if no cup addressing).</td>
<td></td>
</tr>
<tr>
<td><strong>cursor_up</strong></td>
<td>Moves cursor up one line.</td>
<td></td>
</tr>
<tr>
<td><strong>cursor_visible</strong></td>
<td>Makes cursor very visible.</td>
<td></td>
</tr>
<tr>
<td><strong>define char</strong></td>
<td>Define a character in a character set.</td>
<td></td>
</tr>
<tr>
<td><strong>delete_character</strong></td>
<td>Deletes character. (*)</td>
<td></td>
</tr>
<tr>
<td><strong>delete_line</strong></td>
<td>Deletes line. (*)</td>
<td></td>
</tr>
<tr>
<td>Command</td>
<td>Short Form</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>dial_phone</td>
<td>dial</td>
<td>Dial phone number #1.</td>
</tr>
<tr>
<td>dis_status_line</td>
<td>dsl</td>
<td>Disables status line.</td>
</tr>
<tr>
<td>display_clock</td>
<td>dclk</td>
<td>Display time-of-day clock.</td>
</tr>
<tr>
<td>down_half_line</td>
<td>hd</td>
<td>Indicates subscript (forward 1/2 line feed).</td>
</tr>
<tr>
<td>ena_acs</td>
<td>enacs</td>
<td>Enable alternate character set.</td>
</tr>
<tr>
<td>enter_alt_charset_mode</td>
<td>smacs</td>
<td>Starts alternate character set.</td>
</tr>
<tr>
<td>enter_am_mode</td>
<td>smam</td>
<td>Turn on automatic margins.</td>
</tr>
<tr>
<td>enter_blink_mode</td>
<td>blink</td>
<td>Enables blinking.</td>
</tr>
<tr>
<td>enter_bold_mode</td>
<td>bold</td>
<td>Enables bold (extra bright) mode.</td>
</tr>
<tr>
<td>enter_bottom_mode</td>
<td>btml</td>
<td>Starts bottom line mode. This string capability is an aid for drawing tables and is valid only for aixterm and aixterm-m terminal definitions.</td>
</tr>
<tr>
<td>enter_ca_mode</td>
<td>smcup</td>
<td>Begins programs that use cup addressing.</td>
</tr>
<tr>
<td>enter_delete_mode</td>
<td>smdc</td>
<td>Starts delete mode.</td>
</tr>
<tr>
<td>enter_dim_mode</td>
<td>dim</td>
<td>Enables half-bright mode.</td>
</tr>
<tr>
<td>enter_doublewide_mode</td>
<td>swidm</td>
<td>Enable double-wide printing.</td>
</tr>
<tr>
<td>enter_draft_quality</td>
<td>sdrfq</td>
<td>Set draft quality print.</td>
</tr>
<tr>
<td>enter_insert_mode</td>
<td>smir</td>
<td>Starts insert mode.</td>
</tr>
<tr>
<td>enter_italics_mode</td>
<td>sitm</td>
<td>Enable italics.</td>
</tr>
<tr>
<td>enter_leftward_mode</td>
<td>slm</td>
<td>Enable leftward carriage motion.</td>
</tr>
<tr>
<td>Command Name</td>
<td>Description</td>
<td>Abbreviation</td>
</tr>
<tr>
<td>-------------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>enter_1vert_mode</td>
<td>Starts left vertical line mode. This string capability is an aid for drawing tables. Valid only for aixterm and aixterm-m terminal definitions.</td>
<td>lvert</td>
</tr>
<tr>
<td>enter_micro_mode</td>
<td>Enable micro motion capabilities.</td>
<td>smicm</td>
</tr>
<tr>
<td>enter_near_letter_quality</td>
<td>Set near-letter quality print.</td>
<td>snlq</td>
</tr>
<tr>
<td>enter_normal_quality</td>
<td>Set normal quality print.</td>
<td>snrmq</td>
</tr>
<tr>
<td>enter_protected_mode</td>
<td>Enables protected mode.</td>
<td>prot</td>
</tr>
<tr>
<td>enter_reverse_mode</td>
<td>Enables reverse video mode.</td>
<td>rev</td>
</tr>
<tr>
<td>enter_rvert_mode</td>
<td>Starts right vertical line mode. This string capability is an aid for drawing tables and is valid only for aixterm and aixterm-m terminal definitions.</td>
<td>rvert</td>
</tr>
<tr>
<td>enter_secure_mode</td>
<td>Enables blank mode (characters are invisible).</td>
<td>invis</td>
</tr>
<tr>
<td>enter_shadow_mode</td>
<td>Enable shadow printing.</td>
<td>sshm</td>
</tr>
<tr>
<td>enter_standout_mode</td>
<td>Begins standout mode.</td>
<td>smso</td>
</tr>
<tr>
<td>enter_subscript_mode</td>
<td>Enable subscript printing.</td>
<td>ssubm</td>
</tr>
<tr>
<td>enter_superscript_mode</td>
<td>Enable superscript printing.</td>
<td>ssupm</td>
</tr>
<tr>
<td>enter_topline_mode</td>
<td>Starts top line mode. This string capability is an aid for drawing tables and is valid only for aixterm and aixterm-m terminal definitions.</td>
<td>topl</td>
</tr>
<tr>
<td>enter_underline_mode</td>
<td>Starts underscore mode.</td>
<td>smul</td>
</tr>
<tr>
<td>enter_upward_mode</td>
<td>Enable upward carriage motion.</td>
<td>sum</td>
</tr>
<tr>
<td>enter_xon_mode</td>
<td>Turn on xon/xoff handshaking.</td>
<td>smxon</td>
</tr>
<tr>
<td>erase_chars</td>
<td>Erases #1 characters. (G)</td>
<td>ech</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><code>exit_alt_charset_mode</code></td>
<td>Ends alternate character set.</td>
<td></td>
</tr>
<tr>
<td><code>rmacs</code></td>
<td>ae</td>
<td></td>
</tr>
<tr>
<td><code>exit_am_mode</code></td>
<td>Turn off automatic margins.</td>
<td></td>
</tr>
<tr>
<td><code>rmam</code></td>
<td>RA</td>
<td></td>
</tr>
<tr>
<td><code>exit_attribute_mode</code></td>
<td>Disables all attributes.</td>
<td></td>
</tr>
<tr>
<td><code>sgr0</code></td>
<td>me</td>
<td></td>
</tr>
<tr>
<td><code>exit_ca_mode</code></td>
<td>Ends programs that use <code>cup</code> addressing.</td>
<td></td>
</tr>
<tr>
<td><code>rmcup</code></td>
<td>te</td>
<td></td>
</tr>
<tr>
<td><code>exit_delete_mode</code></td>
<td>Ends delete mode.</td>
<td></td>
</tr>
<tr>
<td><code>rmdc</code></td>
<td>ed</td>
<td></td>
</tr>
<tr>
<td><code>exit_doublewide_mode</code></td>
<td>Disable double-wide printing.</td>
<td></td>
</tr>
<tr>
<td><code>rwidm</code></td>
<td>ZQ</td>
<td></td>
</tr>
<tr>
<td><code>exit_insert_mode</code></td>
<td>Ends insert mode.</td>
<td></td>
</tr>
<tr>
<td><code>rmir</code></td>
<td>ei</td>
<td></td>
</tr>
<tr>
<td><code>exit_italics_mode</code></td>
<td>Disable italics.</td>
<td></td>
</tr>
<tr>
<td><code>ritm</code></td>
<td>ZR</td>
<td></td>
</tr>
<tr>
<td><code>exit_leftward_mode</code></td>
<td>Enable rightward (normal) carriage motion.</td>
<td></td>
</tr>
<tr>
<td><code>rlm</code></td>
<td>ZS</td>
<td></td>
</tr>
<tr>
<td><code>exit_micro_mode</code></td>
<td>Disable micro motion capabilities.</td>
<td></td>
</tr>
<tr>
<td><code>micm</code></td>
<td>ZT</td>
<td></td>
</tr>
<tr>
<td><code>exit_shadow_mode</code></td>
<td>Disable shadow printing.</td>
<td></td>
</tr>
<tr>
<td><code>rshm</code></td>
<td>ZU</td>
<td></td>
</tr>
<tr>
<td><code>exit_standout_mode</code></td>
<td>Ends standout mode.</td>
<td></td>
</tr>
<tr>
<td><code>rmsso</code></td>
<td>se</td>
<td></td>
</tr>
<tr>
<td><code>exit_subscript_mode</code></td>
<td>Disable subscript printing.</td>
<td></td>
</tr>
<tr>
<td><code>rsubm</code></td>
<td>ZV</td>
<td></td>
</tr>
<tr>
<td><code>exit_superscript_mode</code></td>
<td>Disable superscript printing.</td>
<td></td>
</tr>
<tr>
<td><code>rsupm</code></td>
<td>ZW</td>
<td></td>
</tr>
<tr>
<td><code>exit_underline_mode</code></td>
<td>Ends underscore mode.</td>
<td></td>
</tr>
<tr>
<td><code>rmul</code></td>
<td>ue</td>
<td></td>
</tr>
<tr>
<td><code>exit_upward_mode</code></td>
<td>Enable downward (normal) carriage motion.</td>
<td></td>
</tr>
<tr>
<td><code>rum</code></td>
<td>ZX</td>
<td></td>
</tr>
<tr>
<td><code>exit_xon_mode</code></td>
<td>Turn off xon/xoff handshaking.</td>
<td></td>
</tr>
<tr>
<td><code>rmxon</code></td>
<td>RX</td>
<td></td>
</tr>
<tr>
<td><code>flash_screen</code></td>
<td>Indicates visual bell (may not move cursor).</td>
<td></td>
</tr>
<tr>
<td><code>flash</code></td>
<td>vb</td>
<td></td>
</tr>
<tr>
<td><code>fixed_pause</code></td>
<td>Pause for 2-3 seconds.</td>
<td></td>
</tr>
<tr>
<td><code>pause</code></td>
<td>PA</td>
<td></td>
</tr>
<tr>
<td><code>flash_hook</code></td>
<td>Flash the switch hook.</td>
<td></td>
</tr>
<tr>
<td><code>hook</code></td>
<td>fh</td>
<td></td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>font_0</td>
<td>Select font 0.</td>
<td></td>
</tr>
<tr>
<td>font_1</td>
<td>Select font 1.</td>
<td></td>
</tr>
<tr>
<td>font_2</td>
<td>Select font 2.</td>
<td></td>
</tr>
<tr>
<td>font_3</td>
<td>Select font 3.</td>
<td></td>
</tr>
<tr>
<td>font_4</td>
<td>Select font 4.</td>
<td></td>
</tr>
<tr>
<td>font_5</td>
<td>Select font 5.</td>
<td></td>
</tr>
<tr>
<td>font_6</td>
<td>Select font 6.</td>
<td></td>
</tr>
<tr>
<td>font_7</td>
<td>Select font 7.</td>
<td></td>
</tr>
<tr>
<td>form_feed</td>
<td>Ejects page (hardcopy terminal). (*)</td>
<td></td>
</tr>
<tr>
<td>from_status_line</td>
<td>Returns from status line.</td>
<td></td>
</tr>
<tr>
<td>goto_window</td>
<td>Go to window #1.</td>
<td></td>
</tr>
<tr>
<td>hangup</td>
<td>Hang-up phone.</td>
<td></td>
</tr>
<tr>
<td>init_1string</td>
<td>Initiates terminal.</td>
<td></td>
</tr>
<tr>
<td>init_2string</td>
<td>Initiates terminal.</td>
<td></td>
</tr>
<tr>
<td>init_3string</td>
<td>Initiates terminal.</td>
<td></td>
</tr>
<tr>
<td>init_file</td>
<td>Identifies file containing long initialization strings.</td>
<td></td>
</tr>
<tr>
<td>init_prog</td>
<td>Locates the program for initialization.</td>
<td></td>
</tr>
<tr>
<td>initialize_color</td>
<td>Initialize the color definition.</td>
<td></td>
</tr>
<tr>
<td>initialize_pair</td>
<td>Initialize color pair.</td>
<td></td>
</tr>
<tr>
<td>insert_character</td>
<td>Inserts character.</td>
<td></td>
</tr>
<tr>
<td>insert_line</td>
<td>Adds new blank line. (*)</td>
<td></td>
</tr>
<tr>
<td>insert_padding</td>
<td>Inserts pad after character inserted. (*)</td>
<td></td>
</tr>
</tbody>
</table>

600 AIX 5L Version 5.3 Files Reference
<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
<th>Key</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>key_a1</td>
<td>Specifies upper left of keypad.</td>
<td>ka1</td>
<td>K1</td>
</tr>
<tr>
<td>key_a3</td>
<td>Specifies upper right of keypad.</td>
<td>ka3</td>
<td>K3</td>
</tr>
<tr>
<td>key_action</td>
<td>Sent by action key.</td>
<td>kact</td>
<td>kJ</td>
</tr>
<tr>
<td>key_b2</td>
<td>Specifies center of keypad.</td>
<td>kb2</td>
<td>K2</td>
</tr>
<tr>
<td>key_backspace</td>
<td>Sent by backspace key.</td>
<td>kbs</td>
<td>kb</td>
</tr>
<tr>
<td>key_beg</td>
<td>Beginning key. KEY_BEG</td>
<td>kbeg</td>
<td>@1</td>
</tr>
<tr>
<td>key_btab</td>
<td>Sent by backtab key. KEY_BTAB</td>
<td>kcbt</td>
<td>kB</td>
</tr>
<tr>
<td>key_c1</td>
<td>Specifies lower left of keypad.</td>
<td>kc1</td>
<td>K4</td>
</tr>
<tr>
<td>key_c3</td>
<td>Specifies lower right of keypad.</td>
<td>kc3</td>
<td>K5</td>
</tr>
<tr>
<td>key_cancel</td>
<td>Cancel key. KEYCANCEL</td>
<td>kcan</td>
<td>@2</td>
</tr>
<tr>
<td>key_catab</td>
<td>Sent by clear-all-tabs key.</td>
<td>ktbc</td>
<td>ka</td>
</tr>
<tr>
<td>key_clear</td>
<td>Sent by clear screen or erase key.</td>
<td>kclr</td>
<td>kC</td>
</tr>
<tr>
<td>key_close</td>
<td>Close key. KEY_CLOSE</td>
<td>kclo</td>
<td>@3</td>
</tr>
<tr>
<td>key_command</td>
<td>Command-request key.</td>
<td>kcmd</td>
<td>@4</td>
</tr>
<tr>
<td>key_command_pane</td>
<td>Command-pane key.</td>
<td>kcpn</td>
<td>@7</td>
</tr>
<tr>
<td>key_copy</td>
<td>Copy key. KEY_COPY</td>
<td>kcpy</td>
<td>@5</td>
</tr>
<tr>
<td>key_create</td>
<td>Create key. KEYCREATE</td>
<td>kcr</td>
<td>@6</td>
</tr>
<tr>
<td>key_ctab</td>
<td>Sent by clear tab key.</td>
<td>kctab</td>
<td>kt</td>
</tr>
<tr>
<td>key_dc</td>
<td>Sent by delete-character key.</td>
<td>kdch1</td>
<td>kD</td>
</tr>
<tr>
<td>key_dl</td>
<td>Sent by delete-line key.</td>
<td>kd1</td>
<td>kL</td>
</tr>
<tr>
<td>key_do</td>
<td>Do request key.</td>
<td>kdo</td>
<td>ki</td>
</tr>
<tr>
<td>key</td>
<td>Description</td>
<td>kc</td>
<td>kd</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------------------------------------------</td>
<td>-------</td>
<td>------</td>
</tr>
<tr>
<td>key_down</td>
<td>Sent by terminal down-arrow key.</td>
<td>kcud1</td>
<td>kd</td>
</tr>
<tr>
<td>key_eic</td>
<td>Sent by rmir or smir in insert mode.</td>
<td>krmir</td>
<td>kM</td>
</tr>
<tr>
<td>key_end</td>
<td>End key. KEY_END</td>
<td>kend</td>
<td>@7</td>
</tr>
<tr>
<td>key_enter</td>
<td>Enter/send (unreliable). KEY_ENTER.</td>
<td>kent</td>
<td>@8</td>
</tr>
<tr>
<td>key_eol</td>
<td>Sent by clear-to-end-of-line key.</td>
<td>kel</td>
<td>kE</td>
</tr>
<tr>
<td>key_eos</td>
<td>Sent by clear-to-end-of-screen key.</td>
<td>ked</td>
<td>kS</td>
</tr>
<tr>
<td>key_exit</td>
<td>Exit key. KEY_EXIT.</td>
<td>kext</td>
<td>@9</td>
</tr>
<tr>
<td>key_f0</td>
<td>Sent by function key F0.</td>
<td>kf0</td>
<td>k0</td>
</tr>
<tr>
<td>key_f1</td>
<td>Sent by function key F1.</td>
<td>kf1</td>
<td>k1</td>
</tr>
<tr>
<td>key_f2</td>
<td>Sent by function key F2.</td>
<td>kf2</td>
<td>k2</td>
</tr>
<tr>
<td>key_f3</td>
<td>Sent by function key F3.</td>
<td>kf3</td>
<td>k3</td>
</tr>
<tr>
<td>key_f4</td>
<td>Sent by function key F4.</td>
<td>kf4</td>
<td>k4</td>
</tr>
<tr>
<td>key_f5</td>
<td>Sent by function key F5.</td>
<td>kf5</td>
<td>k5</td>
</tr>
<tr>
<td>key_f6</td>
<td>Sent by function key F6.</td>
<td>kf6</td>
<td>k6</td>
</tr>
<tr>
<td>key_f7</td>
<td>Sent by function key F7.</td>
<td>kf7</td>
<td>k7</td>
</tr>
<tr>
<td>key_f8</td>
<td>Sent by function key F8.</td>
<td>kf8</td>
<td>k8</td>
</tr>
<tr>
<td>key_f9</td>
<td>Sent by function key F9.</td>
<td>kf9</td>
<td>k9</td>
</tr>
<tr>
<td>key_f10</td>
<td>Sent by function key F10.</td>
<td>kf10</td>
<td>k;</td>
</tr>
<tr>
<td>key_f11</td>
<td>Sent by function key F11.</td>
<td>kf11</td>
<td>F1</td>
</tr>
<tr>
<td>key_f12</td>
<td>Sent by function key F12.</td>
<td>kf12</td>
<td>F2</td>
</tr>
<tr>
<td>key_f13</td>
<td>Sent by function key F13. KEY_F(13)</td>
<td>kf13</td>
<td>F3</td>
</tr>
<tr>
<td>key_f14</td>
<td>Sent by function key F14. KEY_F(14)</td>
<td>kf14</td>
<td>F4</td>
</tr>
<tr>
<td>key_f15</td>
<td>Sent by function key F15. KEY_F(15)</td>
<td>kf15</td>
<td>F5</td>
</tr>
<tr>
<td>key_f16</td>
<td>Sent by function key 16. KEY_F(16)</td>
<td>kf16</td>
<td>F6</td>
</tr>
<tr>
<td>---------</td>
<td>----------------------------------</td>
<td>------</td>
<td>----</td>
</tr>
<tr>
<td>key_f17</td>
<td>Sent by function key 17. KEY_F(17)</td>
<td>kf17</td>
<td>F7</td>
</tr>
<tr>
<td>key_f18</td>
<td>Sent by function key 18. KEY_F(18)</td>
<td>kf18</td>
<td>F8</td>
</tr>
<tr>
<td>key_f19</td>
<td>Sent by function key 19. KEY_F(19)</td>
<td>kf19</td>
<td>F9</td>
</tr>
<tr>
<td>key_f20</td>
<td>Sent by function key 20. KEY_F(20)</td>
<td>kf20</td>
<td>FA</td>
</tr>
<tr>
<td>key_f21</td>
<td>Sent by function key 21. KEY_F(21)</td>
<td>kf21</td>
<td>FB</td>
</tr>
<tr>
<td>key_f22</td>
<td>Sent by function key 22. KEY_F(22)</td>
<td>kf22</td>
<td>FC</td>
</tr>
<tr>
<td>key_f23</td>
<td>Sent by function key 23. KEY_F(23)</td>
<td>kf23</td>
<td>FD</td>
</tr>
<tr>
<td>key_f24</td>
<td>Sent by function key 24. KEY_F(24)</td>
<td>kf24</td>
<td>FE</td>
</tr>
<tr>
<td>key_f25</td>
<td>Sent by function key 25. KEY_F(25)</td>
<td>kf25</td>
<td>FF</td>
</tr>
<tr>
<td>key_f26</td>
<td>Sent by function key 26. KEY_F(26)</td>
<td>kf26</td>
<td>FG</td>
</tr>
<tr>
<td>key_f27</td>
<td>Sent by function key 27. KEY_F(27), 03543</td>
<td>kf27</td>
<td>FH</td>
</tr>
<tr>
<td>key_f28</td>
<td>Sent by function key 28. KEY_F(28)</td>
<td>kf28</td>
<td>FI</td>
</tr>
<tr>
<td>key_f29</td>
<td>Sent by function key 29. KEY_F(29)</td>
<td>kf29</td>
<td>FJ</td>
</tr>
<tr>
<td>key_f30</td>
<td>Sent by function key 30. KEY_F(30)</td>
<td>kf30</td>
<td>FK</td>
</tr>
<tr>
<td>key_f31</td>
<td>Sent by function key 31. KEY_F(31)</td>
<td>kf31</td>
<td>FL</td>
</tr>
<tr>
<td>key_f32</td>
<td>Sent by function key 32. KEY_F(32)</td>
<td>kf32</td>
<td>FM</td>
</tr>
<tr>
<td>key_f33</td>
<td>Sent by function key 33. KEY_F(33)</td>
<td>kf33</td>
<td>FN</td>
</tr>
<tr>
<td>key_f34</td>
<td>Sent by function key 34. KEY_F(34)</td>
<td>kf34</td>
<td>FO</td>
</tr>
<tr>
<td>key_f35</td>
<td>Sent by function key 35. KEY_F(35)</td>
<td>kf35</td>
<td>FP</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>key_f36</td>
<td>Sent by function key 36. KEY_F(36)</td>
<td>kf36</td>
<td>FP</td>
</tr>
<tr>
<td>key_f37</td>
<td>Sent by function key 37. KEY_F(37)</td>
<td>kf37</td>
<td>FQ</td>
</tr>
<tr>
<td>key_f38</td>
<td>Sent by function key 38. KEY_F(38)</td>
<td>kf38</td>
<td>FR</td>
</tr>
<tr>
<td>key_f39</td>
<td>Sent by function key 39. KEY_F(39)</td>
<td>kf39</td>
<td>FS</td>
</tr>
<tr>
<td>key_f40</td>
<td>Sent by function key 40. KEY_F(40)</td>
<td>kf40</td>
<td>FT</td>
</tr>
<tr>
<td>key_f41</td>
<td>Sent by function key 41. KEY_F(41)</td>
<td>kf41</td>
<td>FU</td>
</tr>
<tr>
<td>key_f42</td>
<td>Sent by function key 42. KEY_F(42)</td>
<td>kf42</td>
<td>FV</td>
</tr>
<tr>
<td>key_f43</td>
<td>Sent by function key 43. KEY_F(43)</td>
<td>kf43</td>
<td>FW</td>
</tr>
<tr>
<td>key_f44</td>
<td>Sent by function key 44. KEY_F(44)</td>
<td>kf44</td>
<td>FX</td>
</tr>
<tr>
<td>key_f45</td>
<td>Sent by function key 45. KEY_F(45)</td>
<td>kf45</td>
<td>FY</td>
</tr>
<tr>
<td>key_f46</td>
<td>Sent by function key 46. KEY_F(46)</td>
<td>kf46</td>
<td>FZ</td>
</tr>
<tr>
<td>key_f47</td>
<td>Sent by function key 47. KEY_F(47)</td>
<td>kf47</td>
<td>Fa</td>
</tr>
<tr>
<td>key_f48</td>
<td>Sent by function key 48. KEY_F(48)</td>
<td>kf48</td>
<td>Fb</td>
</tr>
<tr>
<td>key_f49</td>
<td>Sent by function key 49. KEY_F(49)</td>
<td>kf49</td>
<td>Fc</td>
</tr>
<tr>
<td>key_f50</td>
<td>Sent by function key 50. KEY_F(50)</td>
<td>kf50</td>
<td>Fd</td>
</tr>
<tr>
<td>key_f51</td>
<td>Sent by function key 51. KEY_F(51)</td>
<td>kf51</td>
<td>Fe</td>
</tr>
<tr>
<td>key_f52</td>
<td>Sent by function key 52. KEY_F(52)</td>
<td>kf52</td>
<td>Ff</td>
</tr>
<tr>
<td>key_f53</td>
<td>Sent by function key 53. KEY_F(53)</td>
<td>kf53</td>
<td>Fg</td>
</tr>
<tr>
<td>key_f54</td>
<td>Sent by function key 54. KEY_F(54)</td>
<td>kf54</td>
<td>Fi</td>
</tr>
<tr>
<td>Key</td>
<td>Description</td>
<td>Code</td>
<td>Function</td>
</tr>
<tr>
<td>-------</td>
<td>--------------------------------------------------</td>
<td>------</td>
<td>----------</td>
</tr>
<tr>
<td>f55</td>
<td>Sent by function key f55. KEY_F(55)</td>
<td>kf55</td>
<td>Fj</td>
</tr>
<tr>
<td>f56</td>
<td>Sent by function key f56. KEY_F(56)</td>
<td>kf56</td>
<td>Fk</td>
</tr>
<tr>
<td>f57</td>
<td>Sent by function key f57. KEY_F(57)</td>
<td>kf57</td>
<td>Fl</td>
</tr>
<tr>
<td>f58</td>
<td>Sent by function key f58. KEY_F(58)</td>
<td>kf58</td>
<td>Fm</td>
</tr>
<tr>
<td>f59</td>
<td>Sent by function key f59. KEY_F(59)</td>
<td>kf59</td>
<td>Fn</td>
</tr>
<tr>
<td>f60</td>
<td>Sent by function key f60. KEY_F(60)</td>
<td>kf60</td>
<td>Fo</td>
</tr>
<tr>
<td>f61</td>
<td>Sent by function key f61. KEY_F(61)</td>
<td>kf61</td>
<td>Fp</td>
</tr>
<tr>
<td>f62</td>
<td>Sent by function key f62. KEY_F(62)</td>
<td>kf62</td>
<td>Fq</td>
</tr>
<tr>
<td>f63</td>
<td>Sent by function key f63. KEY_F(63)</td>
<td>kf63</td>
<td>Fr</td>
</tr>
<tr>
<td>find</td>
<td>Find key. KEY_FIND</td>
<td>kfind</td>
<td>@0</td>
</tr>
<tr>
<td>help</td>
<td>Help key.</td>
<td>khlp</td>
<td>kq</td>
</tr>
<tr>
<td>home</td>
<td>Sent by home key.</td>
<td>khome</td>
<td>kh</td>
</tr>
<tr>
<td>ic</td>
<td>Sent by insert-character/enter-insert-mode key.</td>
<td>kich1</td>
<td>kl</td>
</tr>
<tr>
<td>il</td>
<td>Sent by insert line key.</td>
<td>kil1</td>
<td>kA</td>
</tr>
<tr>
<td>left</td>
<td>Sent by terminal left-arrow key.</td>
<td>kcub1</td>
<td>kl</td>
</tr>
<tr>
<td>ll</td>
<td>Sent by home-down key.</td>
<td>kll</td>
<td>kH</td>
</tr>
<tr>
<td>mark</td>
<td>Mark key. KEY_MARK</td>
<td>kmrk</td>
<td>%2</td>
</tr>
<tr>
<td>message</td>
<td>Message key. KEY_MESSAGE</td>
<td>kmsg</td>
<td>%3</td>
</tr>
<tr>
<td>move</td>
<td>Move key. KEY_MOVE</td>
<td>kmov</td>
<td>%4</td>
</tr>
<tr>
<td>newline</td>
<td>New-line key.</td>
<td>knl</td>
<td>kn</td>
</tr>
<tr>
<td>key</td>
<td>Description</td>
<td>Code</td>
<td>Value</td>
</tr>
<tr>
<td>-----</td>
<td>-------------</td>
<td>------</td>
<td>-------</td>
</tr>
<tr>
<td>key_next</td>
<td>Next object key. KEY_NEXT</td>
<td>knxt</td>
<td>%5</td>
</tr>
<tr>
<td>key_next_pane</td>
<td>Next-pane key.</td>
<td>knpn</td>
<td>kv</td>
</tr>
<tr>
<td>key_npage</td>
<td>Sent by next-page key.</td>
<td>knp</td>
<td>kN</td>
</tr>
<tr>
<td>key_open</td>
<td>Open key. KEY_OPEN</td>
<td>kopn</td>
<td>%6</td>
</tr>
<tr>
<td>key_options</td>
<td>Options key. KEY_OPTIONS</td>
<td>kopt</td>
<td>%7</td>
</tr>
<tr>
<td>key_ppage</td>
<td>Sent by previous-page key.</td>
<td>kpp</td>
<td>kP</td>
</tr>
<tr>
<td>key_prev_pane</td>
<td>Sent by previous-pane key.</td>
<td>kppn</td>
<td>kV</td>
</tr>
<tr>
<td>key_prev_cmd</td>
<td>Sent by previous-command key.</td>
<td>kpcmd</td>
<td>kp</td>
</tr>
<tr>
<td>key_previous</td>
<td>Previous object key. KEY_PREVIOUS</td>
<td>kprv</td>
<td>%8</td>
</tr>
<tr>
<td>key_print</td>
<td>Print or copy. KEY_PRINT</td>
<td>kprt</td>
<td>%9</td>
</tr>
<tr>
<td>key_quit</td>
<td>Quit key.</td>
<td>kquit</td>
<td>kQ</td>
</tr>
<tr>
<td>key_redo</td>
<td>Redo key. KEY_REDO</td>
<td>krdo</td>
<td>%0</td>
</tr>
<tr>
<td>key_reference</td>
<td>Reference key. KEY_REFERENCE</td>
<td>kref</td>
<td>&amp;1</td>
</tr>
<tr>
<td>key_refresh</td>
<td>Refresh key. KEY_REFRESH</td>
<td>krfr</td>
<td>&amp;2</td>
</tr>
<tr>
<td>key_replace</td>
<td>Replace key. KEY_REPLACE</td>
<td>krpl</td>
<td>&amp;3</td>
</tr>
<tr>
<td>key_restart</td>
<td>Restart key. KEY_RESTART</td>
<td>krst</td>
<td>&amp;4</td>
</tr>
<tr>
<td>key_resume</td>
<td>Resume key. KEY_RESUME</td>
<td>kres</td>
<td>&amp;5</td>
</tr>
<tr>
<td>key_right</td>
<td>Sent by terminal right-arrow key.</td>
<td>kcuf1</td>
<td>kr</td>
</tr>
<tr>
<td>key_save</td>
<td>Save key. KEY_SAVE</td>
<td>ksv</td>
<td>&amp;6</td>
</tr>
<tr>
<td>key</td>
<td>Description</td>
<td>Code</td>
<td>Value</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------------------------------------------------</td>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td>key_sbeg</td>
<td>Shifted beginning key. KEY_SBEG</td>
<td>kBEG</td>
<td>$9</td>
</tr>
<tr>
<td>key_scancel</td>
<td>Shifted cancel key. KEY_SCANCEL</td>
<td>kCAN</td>
<td>$0</td>
</tr>
<tr>
<td>key_scommand</td>
<td>Shifted command key. KEY_SCOMMAND</td>
<td>kCMD</td>
<td>*1</td>
</tr>
<tr>
<td>key_scopy</td>
<td>Shifted copy key. KEY_SCOPY</td>
<td>kCPY</td>
<td>*2</td>
</tr>
<tr>
<td>key_screate</td>
<td>Shifted create key. KEY_SCREATE</td>
<td>kCRT</td>
<td>*3</td>
</tr>
<tr>
<td>key_scroll_left</td>
<td>Scroll left.</td>
<td>kscl</td>
<td>kz</td>
</tr>
<tr>
<td>key_scroll_right</td>
<td>Scroll right.</td>
<td>kscr</td>
<td>kZ</td>
</tr>
<tr>
<td>key_sdc</td>
<td>Shifted delete-character key. KEY_SDC</td>
<td>kDC</td>
<td>*4</td>
</tr>
<tr>
<td>key sdl</td>
<td>Shifted delete-line key. KEY_SDL</td>
<td>kDL</td>
<td>*5</td>
</tr>
<tr>
<td>key_select</td>
<td>Select key.</td>
<td>kslt</td>
<td>*6</td>
</tr>
<tr>
<td>key_send</td>
<td>Shifted end key. KEY_SEND</td>
<td>kEND</td>
<td>*7</td>
</tr>
<tr>
<td>key_seol</td>
<td>Shifted clear-line key. KEY_SEOL</td>
<td>kEOL</td>
<td>*8</td>
</tr>
<tr>
<td>key_sexit</td>
<td>Shifted exit key. KEY_SEXIT</td>
<td>kEXT</td>
<td>*9</td>
</tr>
<tr>
<td>key_sf</td>
<td>Sent by scroll-forward/ scroll-down key.</td>
<td>kind</td>
<td>kF</td>
</tr>
<tr>
<td>key_sf1</td>
<td>Special function key 1.</td>
<td>ksf1</td>
<td>S1</td>
</tr>
<tr>
<td>key_sf2</td>
<td>Special function key 2.</td>
<td>ksf2</td>
<td>S2</td>
</tr>
<tr>
<td>key_sf3</td>
<td>Special function key 3.</td>
<td>ksf3</td>
<td>S3</td>
</tr>
<tr>
<td>key_sf4</td>
<td>Special function key 4.</td>
<td>ksf4</td>
<td>S4</td>
</tr>
<tr>
<td>key_sf5</td>
<td>Special function key 5.</td>
<td>ksf5</td>
<td>S5</td>
</tr>
<tr>
<td>key_sf6</td>
<td>Special function key 6.</td>
<td>ksf6</td>
<td>S6</td>
</tr>
<tr>
<td>key_sf7</td>
<td>Special function key 7.</td>
<td>ksf7</td>
<td>S7</td>
</tr>
<tr>
<td>--------</td>
<td>------------------------</td>
<td>------</td>
<td>----</td>
</tr>
<tr>
<td>key_sf8</td>
<td>Special function key 8.</td>
<td>ksf8</td>
<td>S8</td>
</tr>
<tr>
<td>key_sf9</td>
<td>Special function key 9.</td>
<td>ksf9</td>
<td>S9</td>
</tr>
<tr>
<td>key_sf10</td>
<td>Special function key 10.</td>
<td>ksf10</td>
<td>S0</td>
</tr>
<tr>
<td>key_sfind</td>
<td>Shifted find key. KEY_SFIND</td>
<td>kFND</td>
<td>*0</td>
</tr>
<tr>
<td>key_shelp</td>
<td>Shifted help key. KEY_SHELP</td>
<td>kHLP</td>
<td>#1</td>
</tr>
<tr>
<td>key_shome</td>
<td>Shifted home key. KEY_SHOME</td>
<td>kHOM</td>
<td>#2</td>
</tr>
<tr>
<td>key_sic</td>
<td>Shifted input key. KEY_SIC</td>
<td>kIC</td>
<td>#3</td>
</tr>
<tr>
<td>key_sleft</td>
<td>Shifted left-arrow key. KEY_SLEFT</td>
<td>kLFT</td>
<td>#4</td>
</tr>
<tr>
<td>key_smap_in1</td>
<td>Input for special mapped key 1.</td>
<td>kmpf1</td>
<td>Kv</td>
</tr>
<tr>
<td>key_smap_in2</td>
<td>Input for special mapped key 2.</td>
<td>kmpf2</td>
<td>Kw</td>
</tr>
<tr>
<td>key_smap_in3</td>
<td>Input for special mapped key 3.</td>
<td>kmpf3</td>
<td>Kx</td>
</tr>
<tr>
<td>key_smap_in4</td>
<td>Input for special mapped key 4.</td>
<td>kmpf4</td>
<td>Ky</td>
</tr>
<tr>
<td>key_smap_in5</td>
<td>Input for special mapped key 5.</td>
<td>kmpf5</td>
<td>Kz</td>
</tr>
<tr>
<td>key_smap_in6</td>
<td>Input for special mapped key 6.</td>
<td>kmpf6</td>
<td>Kr</td>
</tr>
<tr>
<td>key_smap_in7</td>
<td>Input for special mapped key 7.</td>
<td>kmpf7</td>
<td>Ks</td>
</tr>
<tr>
<td>key_smap_in8</td>
<td>Input for special mapped key 8.</td>
<td>kmpf8</td>
<td>Kt</td>
</tr>
<tr>
<td>key_smap_in9</td>
<td>Input for special mapped key 9.</td>
<td>kmpf9</td>
<td>Ku</td>
</tr>
<tr>
<td>key_smap_out1</td>
<td>Output for mapped key 1.</td>
<td>kmpt1</td>
<td>KV</td>
</tr>
<tr>
<td>key_smap_out2</td>
<td>kmpt2</td>
<td>KW</td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td>-------</td>
<td>----</td>
<td></td>
</tr>
<tr>
<td>key_smap_out3</td>
<td>kmpt3</td>
<td>KX</td>
<td></td>
</tr>
<tr>
<td>key_smap_out4</td>
<td>kmpt4</td>
<td>KY</td>
<td></td>
</tr>
<tr>
<td>key_smap_out5</td>
<td>kmpt5</td>
<td>KZ</td>
<td></td>
</tr>
<tr>
<td>key_smap_out6</td>
<td>kmpt6</td>
<td>KR</td>
<td></td>
</tr>
<tr>
<td>key_smap_out7</td>
<td>kmpt7</td>
<td>KS</td>
<td></td>
</tr>
<tr>
<td>key_smap_out8</td>
<td>kmpt8</td>
<td>KT</td>
<td></td>
</tr>
<tr>
<td>key_smap_out9</td>
<td>kmpt9</td>
<td>KU</td>
<td></td>
</tr>
<tr>
<td>key_smessage</td>
<td>kMSG</td>
<td>%a</td>
<td></td>
</tr>
<tr>
<td>key_smove</td>
<td>kMOV</td>
<td>%b</td>
<td></td>
</tr>
<tr>
<td>key_snext</td>
<td>kNXT</td>
<td>%c</td>
<td></td>
</tr>
<tr>
<td>key_soptions</td>
<td>kOPT</td>
<td>%d</td>
<td></td>
</tr>
<tr>
<td>key_sprevious</td>
<td>kPRV</td>
<td>%e</td>
<td></td>
</tr>
<tr>
<td>key_sprint</td>
<td>kPRT</td>
<td>%f</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>key_sr</th>
<th>kri</th>
<th>kR</th>
</tr>
</thead>
<tbody>
<tr>
<td>key_redo</td>
<td>kRDO</td>
<td>%g</td>
</tr>
<tr>
<td>key_replace</td>
<td>kRPL</td>
<td>%h</td>
</tr>
<tr>
<td>key_sright</td>
<td>kRIT</td>
<td>%i</td>
</tr>
<tr>
<td>key_srsrsume</td>
<td>kRES</td>
<td>%j</td>
</tr>
<tr>
<td>key_sssave</td>
<td>kSAV</td>
<td>!1</td>
</tr>
<tr>
<td>Key</td>
<td>Description</td>
<td>Code</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>key_ssuspend</td>
<td>Shifted suspend key. KEY_SSUPEND</td>
<td>kSPD</td>
</tr>
<tr>
<td>key_stab</td>
<td>Sent by set-tab key.</td>
<td>khts</td>
</tr>
<tr>
<td>key_sundo</td>
<td>Shifted undo key. KEY_SUNDO</td>
<td>kUND</td>
</tr>
<tr>
<td>key_suspend</td>
<td>Suspend key. KEY_SUSPEND</td>
<td>kspd</td>
</tr>
<tr>
<td>key_tab</td>
<td>Tab key.</td>
<td>ktab</td>
</tr>
<tr>
<td>key_undo</td>
<td>Undo key. KEY_UNDO</td>
<td>kund</td>
</tr>
<tr>
<td>key_up</td>
<td>Sent by terminal up-arrow key.</td>
<td>kcuu1</td>
</tr>
<tr>
<td>keypad_local</td>
<td>Ends keypad transmit mode.</td>
<td>rmkx</td>
</tr>
<tr>
<td>keypad_xmit</td>
<td>Puts terminal in keypad transmit mode.</td>
<td>smkx</td>
</tr>
<tr>
<td>lab_f0</td>
<td>Labels function key F0, if not F0.</td>
<td>I0</td>
</tr>
<tr>
<td>lab_f1</td>
<td>Labels function key F1, if not F1.</td>
<td>I1</td>
</tr>
<tr>
<td>lab_f2</td>
<td>Labels function key F2, if not F2.</td>
<td>I2</td>
</tr>
<tr>
<td>lab_f3</td>
<td>Labels function key F3, if not F3.</td>
<td>I3</td>
</tr>
<tr>
<td>lab_f4</td>
<td>Labels function key F4, if not F4.</td>
<td>I4</td>
</tr>
<tr>
<td>lab_f5</td>
<td>Labels function key F5, if not F5.</td>
<td>I5</td>
</tr>
<tr>
<td>lab_f6</td>
<td>Labels function key F6, if not F6.</td>
<td>I6</td>
</tr>
<tr>
<td>lab_f7</td>
<td>Labels function key F7, if not F7.</td>
<td>I7</td>
</tr>
<tr>
<td>lab_f8</td>
<td>Labels function key F8, if not F8.</td>
<td>I8</td>
</tr>
<tr>
<td>lab_f9</td>
<td>Labels function key F9, if not F9.</td>
<td>I9</td>
</tr>
<tr>
<td>lab_f10</td>
<td>Labels function key F10, if not F10.</td>
<td>I10</td>
</tr>
<tr>
<td>label_format</td>
<td>Label format.</td>
<td>fln</td>
</tr>
<tr>
<td>label_off</td>
<td>Turn off soft labels.</td>
<td>rmln</td>
</tr>
<tr>
<td>label_on</td>
<td>Turn on soft labels.</td>
<td>smln</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><strong>meta_on</strong></td>
<td>Enables meta mode (8th bit).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>smm</td>
<td></td>
</tr>
<tr>
<td><strong>meta_off</strong></td>
<td>Disables meta mode.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>rmm</td>
<td></td>
</tr>
<tr>
<td><strong>micro_column_address</strong></td>
<td>Move N steps from the left.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>mhpa</td>
<td></td>
</tr>
<tr>
<td><strong>micro_down</strong></td>
<td>Move 1 step down.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>mcud1</td>
<td></td>
</tr>
<tr>
<td><strong>micro_left</strong></td>
<td>Move 1 step left.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>mcub1</td>
<td></td>
</tr>
<tr>
<td><strong>micro_right</strong></td>
<td>Move 1 step right.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>mcuf1</td>
<td></td>
</tr>
<tr>
<td><strong>micro_row_address</strong></td>
<td>Move N steps from the top.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>mvpa</td>
<td></td>
</tr>
<tr>
<td><strong>micro_up</strong></td>
<td>Move 1 step up.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>mcuu1</td>
<td></td>
</tr>
<tr>
<td><strong>newline</strong></td>
<td>Performs new-line function (behaves like carriage return followed by line feed).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>nel</td>
<td></td>
</tr>
<tr>
<td><strong>order_of_pins</strong></td>
<td>Matches software bits to print-head pins.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>porder</td>
<td></td>
</tr>
<tr>
<td><strong>orig_colors</strong></td>
<td>Original colors.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>oc</td>
<td></td>
</tr>
<tr>
<td><strong>orig_pair</strong></td>
<td>Original color-pair.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>op</td>
<td></td>
</tr>
<tr>
<td><strong>pad_char</strong></td>
<td>Pads character (instead of NULL).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>pad</td>
<td></td>
</tr>
<tr>
<td><strong>parm_dch</strong></td>
<td>Deletes #1 characters.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>dch</td>
<td></td>
</tr>
<tr>
<td><strong>parm_delete_line</strong></td>
<td>Deletes #1 lines.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>dl</td>
<td></td>
</tr>
<tr>
<td><strong>parm_down_cursor</strong></td>
<td>Moves cursor down #1 lines.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>cud</td>
<td></td>
</tr>
<tr>
<td><strong>parm_down_micro</strong></td>
<td>Move $N$ steps down.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>mcud</td>
<td></td>
</tr>
<tr>
<td><strong>parm_ich</strong></td>
<td>Inserts #1 blank characters.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ich</td>
<td></td>
</tr>
<tr>
<td><strong>parm_index</strong></td>
<td>Scrolls forward #1 lines.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>indn</td>
<td></td>
</tr>
<tr>
<td><strong>parm_insert_line</strong></td>
<td>Adds #1 new blank lines.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>il</td>
<td></td>
</tr>
</tbody>
</table>

Chapter 2. File Formats  611
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>parm_left_cursor</td>
<td>Moves cursor left #1 spaces. (G)</td>
</tr>
<tr>
<td>parm_left_micro</td>
<td>Move N steps left.</td>
</tr>
<tr>
<td>parm_right_cursor</td>
<td>Moves cursor right #1 spaces. (G*)</td>
</tr>
<tr>
<td>parm_right_micro</td>
<td>Move N steps right.</td>
</tr>
<tr>
<td>parm_rindex</td>
<td>Scrolls backward #1 lines. (G)</td>
</tr>
<tr>
<td>parm_up_cursor</td>
<td>Moves cursor up #1 lines. (G*)</td>
</tr>
<tr>
<td>parm_up_micro</td>
<td>Move N steps up.</td>
</tr>
<tr>
<td>pkey_key</td>
<td>Programs function key #1 to type string #2.</td>
</tr>
<tr>
<td>pkey_local</td>
<td>Programs function key #1 to execute string #2.</td>
</tr>
<tr>
<td>pkey_xmit</td>
<td>Programs function key #1 to transmit string #2.</td>
</tr>
<tr>
<td>plab_norm</td>
<td>Program label #1 to show string #2.</td>
</tr>
<tr>
<td>print_screen</td>
<td>Prints contents of the screen.</td>
</tr>
<tr>
<td>prtr_non</td>
<td>Enables the printer for #1 bytes.</td>
</tr>
<tr>
<td>prtr_off</td>
<td>Disables the printer.</td>
</tr>
<tr>
<td>prtr_on</td>
<td>Enables the printer.</td>
</tr>
<tr>
<td>pulse</td>
<td>Select pulse dialing.</td>
</tr>
<tr>
<td>quick_dial</td>
<td>Dial phone number #1, without progress detection.</td>
</tr>
<tr>
<td>remove_clock</td>
<td>Remove time-of-day clock.</td>
</tr>
<tr>
<td>repeat_char</td>
<td>Repeats #1 character #2 times. (G*)</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>req_for_input</td>
<td>Send next input char (for pty's).</td>
</tr>
<tr>
<td>reset_1string</td>
<td>Resets terminal to known modes.</td>
</tr>
<tr>
<td>reset_2string</td>
<td>Resets terminal to known modes.</td>
</tr>
<tr>
<td>reset_3string</td>
<td>Resets terminal to known modes.</td>
</tr>
<tr>
<td>reset_file</td>
<td>Identifies the file containing reset string.</td>
</tr>
<tr>
<td>restore_cursor</td>
<td>Restores cursor to position of last sc (save_cursor).</td>
</tr>
<tr>
<td>row_address</td>
<td>Positions cursor to an absolute vertical position (set row). (G)</td>
</tr>
<tr>
<td>save_cursor</td>
<td>Saves cursor position.</td>
</tr>
<tr>
<td>scroll_forward</td>
<td>Scrolls text up.</td>
</tr>
<tr>
<td>scroll_reverse</td>
<td>Scrolls text down.</td>
</tr>
<tr>
<td>select_char_set</td>
<td>Select character set.</td>
</tr>
<tr>
<td>set_attributes</td>
<td>Defines the video attributes. (G) #1-#9</td>
</tr>
<tr>
<td>set_background</td>
<td>Set background color.</td>
</tr>
<tr>
<td>set_bottom_margin</td>
<td>Set soft bottom margin at current line.</td>
</tr>
<tr>
<td>set_bottom_margin_parm</td>
<td>Set soft bottom margin.</td>
</tr>
<tr>
<td>set_clock</td>
<td>Set time-of-day clock.</td>
</tr>
<tr>
<td>set_color_pair</td>
<td>Set color pair.</td>
</tr>
<tr>
<td>set_foreground</td>
<td>Set foreground color.</td>
</tr>
<tr>
<td>set_left_margin</td>
<td>Set soft left margin.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>set_left_margin_parm</code></td>
<td>Set soft left margin.</td>
</tr>
<tr>
<td><code>set_right_margin</code></td>
<td>Set soft right margin.</td>
</tr>
<tr>
<td><code>set_right_margin_parm</code></td>
<td>Set soft right margin.</td>
</tr>
<tr>
<td><code>set_tab</code></td>
<td>Sets a tab in every row of the current column.</td>
</tr>
<tr>
<td><code>set_top_margin</code></td>
<td>Set top margin at current line.</td>
</tr>
<tr>
<td><code>set_top_margin_parm</code></td>
<td>Set soft top margin.</td>
</tr>
<tr>
<td><code>set_window</code></td>
<td>Indicates current window is lines #1-#2, columns #3-#4. (G)</td>
</tr>
<tr>
<td><code>start_bit_image</code></td>
<td>Start printing bit-image graphics.</td>
</tr>
<tr>
<td><code>start_char_set_def</code></td>
<td>Start definition of a character set.</td>
</tr>
<tr>
<td><code>stop_bit_image</code></td>
<td>End printing bit image graphics.</td>
</tr>
<tr>
<td><code>stop_char_set_def</code></td>
<td>End definition of a character set.</td>
</tr>
<tr>
<td><code>subscript_characters</code></td>
<td>List of characters that can appear in subscript.</td>
</tr>
<tr>
<td><code>superscript_characters</code></td>
<td>List of characters that can appear in superscript.</td>
</tr>
<tr>
<td><code>tab</code></td>
<td>Tabs to next 8-space hardware tab stop.</td>
</tr>
<tr>
<td><code>these_cause_cr</code></td>
<td>Printing any of these characters cause a carriage return.</td>
</tr>
<tr>
<td><code>tone</code></td>
<td>Select touch-tone dialing.</td>
</tr>
<tr>
<td><code>to_status_line</code></td>
<td>Moves to status line, column #1. (G)</td>
</tr>
<tr>
<td><code>underline_char</code></td>
<td>Underscores one character and moves beyond it.</td>
</tr>
<tr>
<td><code>up_half_line</code></td>
<td>Indicates superscript (reverse 1/2 line-feed)</td>
</tr>
<tr>
<td><code>user0</code></td>
<td>User string 0.</td>
</tr>
<tr>
<td>user1</td>
<td>User string 1.</td>
</tr>
<tr>
<td>-------</td>
<td>----------------</td>
</tr>
<tr>
<td>user2</td>
<td>User string 2.</td>
</tr>
<tr>
<td>user3</td>
<td>User string 3.</td>
</tr>
<tr>
<td>user4</td>
<td>User string 4.</td>
</tr>
<tr>
<td>user5</td>
<td>User string 5.</td>
</tr>
<tr>
<td>user6</td>
<td>User string 6.</td>
</tr>
<tr>
<td>user7</td>
<td>User string 7.</td>
</tr>
<tr>
<td>user8</td>
<td>User string 8.</td>
</tr>
<tr>
<td>user9</td>
<td>User string 9.</td>
</tr>
<tr>
<td>wait_tone</td>
<td>Wait for dial tone.</td>
</tr>
<tr>
<td>xoff_character</td>
<td>X-off character.</td>
</tr>
<tr>
<td>xon_character</td>
<td>X-on character.</td>
</tr>
<tr>
<td>zero_motion</td>
<td>No motion for the subsequent character.</td>
</tr>
</tbody>
</table>

### Preparing Descriptions

You can create a terminal description by copying and then modifying the description of a similar terminal. You can check the accuracy of your partial descriptions with the vi editor. Some terminals may reveal bugs in the vi editor as well as deficiencies in the ability of the terminfo database to provide a terminal description.

To test a new terminal description, set the `TERMINFO` environment variable to the path name of the directory containing the compiled description on which you are working. Programs then check that directory instead of the `/usr/share/lib/terminfo` directory.

To test for correct padding (if known), do the following:

1. Edit the `/etc/passwd` file at 9600 baud.
2. Delete about 16 lines from the middle of the screen.
3. Press the u key several times quickly.
   - If the terminal fails to display the result properly, more padding is usually needed. You can perform a similar test for insert character.

   **Note:** Excessive padding slows down the terminal.

### Basic Capabilities

This section describes some basic terminal capabilities. If a terminal supports one of these capabilities, the terminal’s terminfo source file entry indicates it. The following list is a list of basic capabilities:
am Indicates that the cursor moves to the beginning of the next line when it reaches the right margin. This capability also indicates whether the cursor can move beyond the bottom right corner of the screen.

bel Produces an audible signal (such as a bell or a beep).

bw Indicates that a backspace from the left edge of the terminal moves the cursor to the last column of the previous row.

clear Clears the screen, leaving the cursor in the home position.

cols Specifies the number of columns on each line for the terminal.

cr Moves the cursor to the left edge of the current row. This code is usually carriage return (Ctrl-M).

cub1 Moves the cursor one space to the left, such as backspace.

cuf1 Moves the cursor to the right one space.

cuu1 Moves the cursor up one space.

cud1 Move the cursor down one space.

hc Specifies a printing terminal with no softcopy unit. You should also specify the os capability.

ind Scrolls text up.

If Specifies a line-feed.

lines Specifies the number of lines on a cathode ray tube (CRT) terminal.

nel Specifies a new-line. The terminal behaves as if it received a carriage return followed by a line feed.

os Indicates that when a character is displayed or printed in a position already occupied by another character, the terminal overstrikes the existing character, rather than replacing it with the new character. The os capability applies to storage scope, printing, and APL terminals.

ri Scrolls text down.

If the LINES and COLUMNS environment variables are set, these variables override the values in the terminfo database.

The local cursor motions encoded in the terminfo database files are undefined at the left and top edges of a CRT terminal. Programs should never attempt to backspace around the left edge (unless the bw string is given) or to go up locally off the top.

To scroll text up, a program should go to the bottom left corner of the screen and send the index string. To scroll text down, a program goes to the top left corner of the screen and sends the reverse index string. The index string is specified by the ind capability and the reverse index string is specified by the ri capability. The index string and the reverse index string are undefined when not on their respective corners of the screen.

The am capability determines whether the cursor sticks at the right edge of the screen when text is output, but this does not necessarily apply when the cursor is moved to the right (the cuf1 capability) from the last column. A terminal has local motion from the left edge only if the bw capability is defined. The cursor then goes to the right edge of the previous row when moved to the left (the cub1 capability) from the left edge. If the terminal does not have the bw capability, the effect is undefined, which is useful for drawing a box around the edge of the screen, for example.

A terminal has switch-selectable automatic margins if the am capability is specified. If the terminal has a command that moves to the first column of the next line, you can define the nel (new-line) capability. It does not matter whether the command clears the remainder of the current line. Therefore, if the terminal has no cr and If, a working nel can still be crafted out of one or both of them.

These capabilities suffice to describe printing terminals and simple CRT terminals. Thus, the Model 33 Teletype is described as:

```
33 | tty33 | tty | Model 33 Teletype
  bel=^G, cols#72, cr=^M, cud1=^J, hc, ind=^J, os, xon,
```

Another terminal is described as:
Parameterized Strings

Cursor-addressing and other strings requiring parameters are described by parameterized string capabilities. These strings have escape sequences similar to the `printf %x` format. For example, to address the cursor, you specify the `cup` capability using the row and column parameters.

The parameterized capabilities include:

- **cub1**: Backspaces the cursor one space.
- **cup**: Addresses the cursor using the row and column parameters. Rows and columns are numbered starting with 0 and refer to the physical screen visible to the user, not to memory.
- **hpa** and **vpa**: Indicates the cursor has row or column absolute cursor addressing: horizontal position absolute (**hpa**) or vertical absolute (**vpa**).

Sometimes the **hpa** and **vpa** capabilities are shorter than the more general two-parameter sequence and you can use them in preference to the **cup** capability. Parameterized local motions (such as, a move of n spaces to the right) are defined with the **cud**, **cub**, **cuf**, and **cuu** capabilities, with a single parameter indicating how many spaces to move. These capabilities are primarily useful if the terminal does not have **cup** capability.

- **ind** and **rin**: Scrolls text. These are parameterized versions of the basic **ind** and **rin** capabilities. The n value is a number of lines.
- **mrcup**: Indicates the terminal has memory-relative cursor addressing.

The parameter mechanism uses a stack and has special % (percent sign) codes to manipulate the stack. Typically, a sequence pushes one of the parameters onto the stack and then prints it in some format. Often, more complex operations are necessary. The encodings have the following meanings:

- **%%**: Outputs a % (percent sign).
- **%[ [:] Flags] [Width [.Precision] ] [doxXs]**: As in the `printf` command, flags are the [± + #] and space.
- **%d**: Prints pop() as in the `printf` command (numeric string from stack).
- **%2d**: Prints pop() like `%2d` (minimum 2 digits output from stack).
- **%3d**: Prints pop() like `%3d` (minimum 3 digits output from stack).
- **%02d**: Prints as in the `printf` command (2 digits output).
- **%03d**: Prints as in the `printf` command (3 digits output).
- **%c**: Prints pop() gives `%c` (character output from stack).
- **%s**: Prints pop() gives `%s` (string output from stack).
- **%p[i]**: Pushes the **i**th parameter onto the stack where **i** is a number between 1 and 9.
- **%P[a-z]**: Sets variable [a-z] to pop() (variable output from stack).
- **%`c`**: Character constant c.
- **%{nn}**: Integer constant nn.
- **%i**: Push strlen (pop( )).
- **%+ %- %* %f %m**: Arithmetic operators (%m is modulus): push (pop() operation pop()).
- **%& %| %^**: Bit operations: push (pop() operation pop()).
- **%== %>> %<**: Logical operations: push (pop() operation pop()).
- **%! %-**: Unary operations: push (operation pop()).
- **%i**: Add 1 to first two parameters (for ANSI terminals).
%?expr %t thenpart %e elsepart %;

If-then-else. The %e elsepart is optional. You can make an else-if construct as with Algol 68 in the following example, where ci denotes conditions and bi bodies.
%? c1 %t b1 %e c2 %t b2 %e c3 %t b3 %e b4 %;

Binary operations are in postfix form with the operands in the usual order. That is, to get x - 5 use %gx{5}%-.

If you use the - (minus sign) flag with %[doxs], then you must place a colon between the % (percent sign) and the - (minus sign) to differentiate the flag from the %- binary operation, for example, %-16.16s.

Consider a terminal that needs to be sent \E&a12c03Y padded for 6 milliseconds to get to row 3 and column 12. Here the order of the rows and columns is inverted, and the row and column are zero-padded as two digits. Thus, the cup capability of this terminal is cup=\E&a%p2%2.2dc%p1%2.2dY$<6>.

Some terminals need the current row and column sent, preceded by a ^T, with the row and column encoded in binary: cup=^T%p1%c%p2%c. Terminals that use %c need to be able to backspace the cursor (cub1) and to move the cursor up one line on the screen (cuu1). This is necessary because it is not always safe to transmit \n, ^D, and ^r characters, since the system may change or discard them.

Note: The library routines dealing with the terminfo database files set terminal modes so that tabs are not expanded by the operating system; thus, \t (tab) is safe to send.

A final example is a terminal that uses row and column offset by a blank character: cup=\E=%p1%’%s%’+%c %p2%’%s%’+%c. After sending \E=, this operation pushes the first parameter, pushes the ASCII value for a space (32), adds them (pushing the sum on the stack in place of the two previous values), and outputs that value as a character. Then the same is done for the second parameter. More complex arithmetic is possible using the stack.

Cursor Motions
The top left corner of the screen is the home position. If the terminal has a fast way to get the cursor to the home position, specify the home capability. Specify, a fast way of getting to the bottom left corner with the ll capability. This method may involve going up (cuu1) from the home position, but a program should never do this itself (unless ll does) because the effect of moving up from the home position is not certain.

Note: The home position is the same as addressing (0,0) to the top left corner of the screen, not of memory.

If the terminal has row or column absolute-cursor addressing, you should specify the single hpa capability (horizontal position above) and the vpa capability (vertical position absolute). Sometimes these are shorter than the more general two parameter sequence and you can use them instead of the cup capability.

If the terminal has parameterized local motions for example, it is capable of moving the cursor n spaces right, you can specify the cud, cub, cuf, and cuu capabilities with a single parameter indicating how many spaces to move. These capabilities are useful if the terminal does not have the cup capability.

Area Clears
The following capabilities clear large areas of the terminal:

- ed: Clears from the current position to the end of the display. This is defined only from the first column of a line. (Thus, it can be simulated by a request to delete a large number of lines, if a true ed is not available.)
- el: Clears from the current cursor position to the end of the line without moving the cursor.
- el1: Clears from the beginning of the line to the current position, inclusive. The cursor is not moved.
Scrolling
The following insert-line and delete-line capabilities are used to indicate a terminal can:

csr  Change the scrolling region. This capability takes two parameters: the top and bottom lines of the scrolling region. The top line of the screen is 0. After using this capability, the cursor position is undefined. See the sc and rc capabilities in this section.
da  Retain the display above the screen. If a line is deleted or the screen is scrolled, non-blank lines can be brought in at the top. This capability is usually defined for multipage terminals.
db  Retain the display below the screen. If a line is deleted or the screen is reverse scrolled, the terminal can bring the non-blank lines at the bottom. This capability is usually defined for multipage terminals.
dl1  Delete the line the cursor is on. This is done only from the first position on the line to be deleted. Additionally, the dl capability takes a single parameter indicating the number of lines to be deleted.
il1  Create a new blank line before the line where the cursor is currently located and scrolls the rest of the screen down. This is done only from the first position of a line. The cursor then appears on the newly blank line. Additionally, the il capability can take a single parameter indicating the number of lines to insert.
ind  Index or scroll forward. A terminal with this capability can shift the display up one line by deleting the top line and adding a blank line at the bottom.
indn Specify the number of lines to scroll forward. This capability has meaning only if the ind capability is also defined.
rc  Restore the cursor. This capability is useful with the csr and sc capabilities.
ri  Reverse scrolling. With this capability, the terminal can shift the screen down by deleting the bottom line and adding a blank line at the top.
rin  Specify the number of lines to reverse scroll. This capability has meaning only if the ri capability also is defined.
sc  Save the cursor. If defined, you can use the sc capability to save the cursor before using the csr capability. Saving the cursor is necessary because the cursor position is undefined after you use the csr capability. Use the rc capability to restore the cursor to the position it held before you used the csr capability.
wind Indicates the terminal has the ability to define a window as part of memory. This is a parameterized string capability with four parameters: the starting and ending lines in memory and the starting and ending columns in memory, in that order.

A terminal that has the csr capability can scroll part of its screen while leaving other lines above and below the region untouched. A forward scroll applied to a region deletes the top of the region, shifts, and adds a line to the bottom of the region. When finished with the scrolling region, you should use the csr capability to restore the scrolling region to the full screen.

Be sure you move the cursor into the scrolling region with the cup capability before you attempt to scroll the region. You should not move the cursor from the region until you are done with it.

Note: If you are using a terminals csr capability, you may also need to use the sc and rc capability.

Terminals that have csr defined have a destructive scrolling region. Once a line is scrolled off the screen, the terminal cannot retrieve it. A terminal with a non-destructive scrolling region can restore scrolled lines by reversing the scrolling. Unless the ind, ri, indn, rin, dl, and dl1 all simulate destructive scrolling, do not specify the csr capability if the terminal has non-destructive scrolling regions.

On multipage terminals, scrolling can put a line onto another page and scrolling in the opposite direction brings the line back. Similarly, deleting a line can cause a line from another page to appear on the screen. Multipage terminals should have the da and db capabilities defined so that programs that use scrolling can adjust their behavior.

A few terminals can define a window as part of memory. For these types of terminals, all clearing, deletion, insertion, and wrapping commands affect the area in memory where the window is defined.
Insert or Delete Character

Generally, terminals handle insert/delete character operations in one of two ways. The most common insert/delete character operations affect only the characters on the current line and shift characters to the right and off the line. Other terminals make a distinction between typed and untyped blanks on the screen. When inserting a character, the displayed data is shifted and an untyped blank is eliminated. Once all the untyped blanks are eliminated, the displayed data wraps to the next line if you continue to insert characters. When deleting a character, an untyped blank is added to the line to compensate for the deleted character.

Generally, terminals insert/delete characters in one-line mode or multiline mode. The two types of terminals also handle untyped spaces differently. One-line mode is the most common mode. In one-line mode, insert/delete character operations affect only the characters on the current line. Insertions shift characters to the right and off the line.

Multiline mode terminals can affect more than one line. In this mode, the terminal makes a distinction between typed and untyped blanks on the screen. Inserting a character on a multiline mode terminal shifts the displayed data and eliminates untyped blanks. If all the untyped blanks are eliminated and you continue to insert characters, the display wraps to the next line. When deleting a character, multiline terminals add an untyped blank to the line to compensate for the deleted character.

Determining Your Terminal’s Type

Clearing a screen and then typing text separated by cursor motions helps you determine the type of insert/delete operations your terminal performs. Clear the screen, then proceed as follows:

1. Type abc def using local cursor movements, not spaces, between the abc and the def.
2. Position the cursor before the abc.
3. Place the terminal in insert mode.
4. Type a line of text. If your typing causes the abc def characters to shift right and exit the right side of the display, the terminal does not distinguish between blanks and untyped positions.
   
   If the abc moves to positions to the immediate left of the def and the characters move to the right on the line, around the end, and to the next line, the terminal is the second type. This is described by the in capability, which signifies insert null.

Although these two attributes (one-line versus multiline insert mode, and different treatment of untyped spaces) are logically separate, there are no known terminals whose insert mode cannot be described with a single attribute.

Insert or Delete Character Capabilities

The terminfo database describes terminals that have an insert mode as well as terminals that send a simple sequence to open a blank position on the current line. The following are used to describe insert/delete character capabilities:

- dch1: Deletes a single character. The dch capability with one parameter, n, deletes n characters.
- ech: Replaces the specified number of characters, starting at the cursor, with blanks. The cursor position remains unchanged.
- ich1: Opens a space in a line for a character to be inserted. This sequence precedes the actual character insertion. Terminals with a true insert mode would not use this capability.
- ip: Indicates post-padding needed. This is given as a number of milliseconds. Any other sequence that may need to be sent after inserting a single character can be given in this capability.
- mir: Allows cursor movement while in insert mode. It is sometimes necessary to move the cursor while in insert mode to delete characters on the same line. Some terminals may not have this capability due to their handling of insert mode.
- rmdc: Exits delete mode.
- rmir: Ends insert mode.
- rmp: Indicates that padding is necessary between characters typed while not in insert mode. This capability is used in replace mode.
Enters delete mode.
Begins insert mode.

If you are creating a `terminfo` description for a terminal that requires an insert mode and also needs a special code to precede each inserted character, then define the `smir/rmr`, and `ich1` capabilities. The `ich` capability, with the one parameter `n`, opens up `n` spaces so that `n` characters can be inserted.

**Highlighting, Underlining, and Visual Bells**

If your terminal has one or more kinds of display attributes, such as highlighting, underlining, and visual bells, you can present these in a number of ways. Highlighting, such as standout mode, presents a high-contrast, easy-to-read format that adds emphasis to error messages and other important messages. Underlining is another method to focus attention on a particular portion of the terminal. Visual bells include methods such as flashing the screen. The following capabilities describe highlighting, underlining, and visual bells:

- **blink** Indicates terminal has blink highlighting mode.
- **bold** Indicates terminal has extra bright highlighting mode.
- **civis** Makes the cursor invisible.
- **cnorm** Displays a normal cursor. This capability reverses the effects of the `civis` and `cvvis` capabilities.
- **cvvis** Makes the cursor more visible than normal when it is not on the bottom line.
- **dim** Indicates the terminal has half-bright highlighting modes.
- **eo** Indicates that blanks erase overstrikes.
- **enacs** Specifies a command string that enables alternate character set mode. Some terminals cannot enter alternate character set mode without first receiving a specific command. The `enacs` capability defines the command.
- **flash** Indicates the terminal has a way of making the screen flash (as a bell replacement) for errors, without moving the cursor.
- **invis** Indicates the terminal has blanking or invisible-text highlighting modes.
- **msgr** Indicates it is safe to move the cursor in standout mode. Otherwise, programs using standout mode should exit this mode before moving the cursor or sending a new-line. Some terminals automatically leave standout mode when they move to a new line or when the cursor is addressed.
- **nrrmc** Indicates that the `smcup` sequence does not restore the screen after a `rmcup` sequence is output. This means that you cannot restore the screen to the state prior to outputting `rmcup`.
- **os** Indicates the terminal can overstrike an existing character without erasing the original. Overstriking creates a compound character.
- **prot** Indicates the terminal has protected text mode. This means the terminal protects the text from overwriting or erasing. The method of protection is terminal dependent.
- **rev** Indicates the terminal has reverse-video mode.
- **rmacs** Exits the alternate character set mode.
- **rmso** Exits standout mode.
- **rmul** Ends underlining.
- **sgr** Provides a sequence to set arbitrary combinations of attributes. The `sgr` capability can set nine attributes. In order, these attributes are the following:
  - standout
underline
• blink
• dim
• bold
• blank
• protect
• alternate character set

To turn a mode on, set it to a nonzero value. To turn a mode off, set it to 0. The sgr capability can only support those modes for which separate capabilities already exist on the terminal.

sgr0  Turns of all the special modes, including the alternate character set.

smacs
Enters the alternate character set mode.

smcup and rmcup
Indicate the terminal must be in a special mode when running a program that uses any of the highlighting, underlining, or visual bell capabilities. The smcup capability enters this mode, and the rmcup capability exits this mode.

This need arises, for example, with terminals having more than one page of memory. If the terminal has only memory-relative cursor addressing, and not screen-relative cursor addressing, a screen-sized window must be fixed into the terminal for cursor addressing to work properly. This is also used when the smcup capability sets the command character to be used by the terminfo database file.

smso  Enters standout mode.

smul  Begins underlining.

uc    Underlines the current character and moves the cursor one space to the right.

ul    Indicates the terminal correctly generates underlined characters (with no special codes needed), even though it does not overstrike.

xmc   Indicates the number of blanks left if the capability to enter or exit standout mode leaves blank spaces on the screen.

Highlighting, Overstriking, and Underlining
You should choose one display method as standout mode and use it to highlight error messages and other kinds of text to which you want to draw attention. For example, you could choose reverse-video plus half-bright or reverse-video alone. The sequences to enter and exit standout mode are given by the smso and rmso capabilities. If the code to change into or out of standout mode leaves one or even two blank spaces on the screen, then xmc should be given to tell how many spaces are left.

You should specify the ul boolean capability if your terminal generates underlined characters by using the underline character with no special codes. You should specify this capability even if the terminal does not otherwise overstrike characters. For terminals where a character overstriking another leaves both characters on the screen, specify the os capability. If the terminal can erase overstrikes with a blank, then indicate this by specifying the eo capability.

Example of Using the sgr Capability
The following example demonstrates how to use the sgr capability to turn on various modes. Assume that you must define a terminal that requires the following escape sequences to turn on various modes:

<table>
<thead>
<tr>
<th>Terminfo Parameter</th>
<th>Mode</th>
<th>Escape Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>none</td>
<td>\E[0m</td>
</tr>
<tr>
<td>p2</td>
<td>standout</td>
<td>\E[0;4;7m</td>
</tr>
<tr>
<td></td>
<td>underline</td>
<td>\E[0;3m</td>
</tr>
</tbody>
</table>
### Terminfo Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Mode</th>
<th>Escape Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>p3</td>
<td>reverse</td>
<td>\E[0;4m</td>
</tr>
<tr>
<td>p4</td>
<td>blink</td>
<td>\E[0;5m</td>
</tr>
<tr>
<td>p5</td>
<td>dim</td>
<td>\E[0;7m</td>
</tr>
<tr>
<td>p6</td>
<td>bold</td>
<td>\E[0;3;4m</td>
</tr>
<tr>
<td>p7</td>
<td>invis</td>
<td>\E[0;8m</td>
</tr>
<tr>
<td>p8</td>
<td>protect</td>
<td>not available</td>
</tr>
<tr>
<td>p9</td>
<td>altcharset</td>
<td>^O (off) ^N (on)</td>
</tr>
</tbody>
</table>

**Note:** Each escape sequence requires a 0 to turn off other modes before turning on its own mode.

You can simulate some modes by combining others. In this example, the standout attribute escape sequence is a combination of the reverse and dim sequences. Also, in the example the bold sequence is a combination of the reverse and underline sequences. To combine such modes as underline and blink, the sequence to use would be \E[0;3;5m.

You cannot simulate certain modes by combining others. For example, you cannot simulate the protect mode. In this example, the system ignores the p8 parameter. The altcharset mode is different in that it is either ^O or ^N, depending on whether the alternate character mode set is on or off. If all modes were turned on, the sequence would appear as \E[0;3;4;5;7;8m^N.

Some sequences are outputted for one or more modes. For example, the ;3 is outputted when either the p2 parameter or p6 parameter is true. If you write out the above sequences along with their dependencies, the result is the following:

<table>
<thead>
<tr>
<th>Sequence</th>
<th>When To Output</th>
<th>terminfo Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>\E[0</td>
<td>always</td>
<td>\E[0</td>
</tr>
<tr>
<td>;3</td>
<td>if p2 or p6</td>
<td>%?%p2%6%</td>
</tr>
<tr>
<td>;4</td>
<td>if p1 or p3 or p6</td>
<td>%?%p1%3%</td>
</tr>
<tr>
<td>;5</td>
<td>if p4</td>
<td>%?%p4%t;5%;</td>
</tr>
<tr>
<td>;7</td>
<td>if p1 or p5</td>
<td>%?%p1%p5%</td>
</tr>
<tr>
<td>;8</td>
<td>if p7</td>
<td>%?%p1%t;8%;</td>
</tr>
<tr>
<td>m</td>
<td>always</td>
<td>m</td>
</tr>
<tr>
<td>^N or ^O</td>
<td>if p9 ^N, else ^O</td>
<td>%?%p9%t^N%e^O%;</td>
</tr>
</tbody>
</table>

The final result would produce a sgr sequence that appears as follows:

```
sgr=\E[0%?%p2%6%|t;3%;%?%p1%3%|p6%|t;4%;%?%p4%t;5%;%?%p1%p5%|t;7%;%?%p1%t;8%;%?%p9%t^N%e^O%;,
```

### Keypad

If the terminal has a keypad that transmits codes when the keys are pressed, you can define this in the terminfo entry for the terminal. It is not possible to handle terminals where the keypad only works in local mode. If the keypad can be set to transmit or not transmit, give these codes as smkx and rmkx. Otherwise, the keypad is assumed to always transmit.

To define the codes sent by certain other special keys, you can specify the labels with the kf0, kf1, ..., kf10 capabilities. The codes transmitted by certain other special keys can be defined with:

- **kbs** Backspace key.
- **kclr** Clear-screen or erase key.
- **kctab** Clear the tab stop in this column.
kdch1  Delete-character key.
kdl1   Delete-line key.
ked    Clear to end of screen.
kel    Clear to end of line.
khts   Set a tab stop in this column.
kich1  Insert character or enter insert mode.
kl1    Insert line.
kind   Scroll forward or down, or both.
kll    Home down key (home is the lower left corner of the display, in this instance).
krmir  Exit insert mode.
kn     Next page.
kppp   Previous page.
ktbc   Clear-all-tabs key.
ri     Scroll backward or up, or both.

In addition, if the keypad has a three-by-three array of keys including the four arrow keys, specify the other five keys as ka1, ka3, kb2 kc1, and kc3. These keys are useful when you need the effects of a three-by-three directional pad.

Strings that program function keys can be given as the pfkey, pfloc, and pfx capabilities. A string to program the soft screen labels can be given as pln. Each of these strings takes two parameters: the function key number to program (from 0 to 10) and the string with which to program it. Function key numbers out of this range can program undefined keys in a terminal-dependent manner. The capabilities differ in that pfkey causes pressing a given key to be the same as the user typing the given string, pfloc causes the string to be executed by the terminal in local mode, and pfx causes the string to be transmitted to the computer. The capabilities niab, lw, and lh define the number of soft labels and the width and height. Use smln and rmln to specify the commands for turning on and off soft labels. smln is normally output after one or more pln sequences to ensure the change becomes visible.

Tabs and Initialization
If the terminal has hardware tabs, you can use ht capability (usually Ctrl-I) to specify the command to advance to the next tab stop. To specify the command to move left toward the previous tab stop, use the cbt capability. By convention, if the terminal modes indicate that operating system is expanding the tabs rather than sending them to the terminal, programs should not use the ht or cbt capabilities even if they are present, since the user may not have the tab stops properly set.

If the terminal has hardware tabs that are initially set every \( n \) spaces when the terminal is powered up, its terminfo description should define the numeric capability it to show the number of spaces the tabs are set to. Normally, the put init command uses the it parameter to determine whether to set the mode for hardware tab expansion and whether to set the tab stops. If the terminal has tab stops that can be saved in nonvolatile memory, the terminfo description can assume that they are properly set.

Other, similar capabilities include the is1, is2, and is3 initialization strings for the terminal; the iprog capability that specifies the terminal’s initialization program, and the if capability that identifies the name of a file containing long initialization strings. These strings are expected to set the terminal into modes consistent with the rest of the terminfo file description. They are normally sent to the terminal by the put init command each time the user logs in. When the user logs in, the system does the following:
- Runs the iprog program.
- Prints is1.
- Print is2.
- Sets the margins using the mgc, smgl, and smgr capabilities.
- Sets the tabs using tbc and hts capabilities.
- Prints the if file.
• Prints \textit{is3}.

You can set up special terminal modes without duplicating strings by putting the common sequences in the \textit{is2} capability and special cases in the \textit{is1} and \textit{is3} capabilities. To specify sequences that do a harder reset from a totally unknown state, specify the \textit{rs1}, \textit{rs2}, \textit{rs3}, and \textit{rf} capabilities that are the same as \textit{is1}, \textit{is2}, \textit{is3}, and the \textit{if} capabilities.

A few terminals use the \textit{if} and \textit{rf} files. However, the recommended method is to use the initialization and reset strings. These strings are output by the \texttt{tput reset} command. This command is used when the terminal starts behaving strangely or is not responding at all. Commands are normally placed in the \textit{rs1}, \textit{rs2}, \textit{rs3} and \textit{rf} capabilities only if they produce annoying effects on the screen and are not necessary when logging in. For example, the command to set the terminal into 80-column mode would normally be part of \textit{is2}, but it causes an annoying screen behavior and is not necessary since most terminals initialize in 80-column mode.

If there are commands to set and clear tab stops, specify them using the \texttt{tbc} (clear all tab stops) and the \texttt{hts} (set a tab stop in the current column of every row) capabilities. If a more complex sequence is needed to set the tabs, the place the sequence in the \textit{is2} or the \textit{if} capability.

The \texttt{mgc} capability can clear any margin. For more information about how to set and clear margins, see \texttt{Margins}.

\section*{Miscellaneous Strings}

If the terminal requires a character other than a null character as a pad, then specify the \texttt{pad} string. Only the first character of the \texttt{pad} string is used. If a terminal does not have a pad character, specify the \texttt{npc} capability.

If the terminal can move up or down half a line, define the \texttt{hu} (half-line up) and \texttt{hd} (half-line down) capabilities. These capabilities are primarily useful for superscripts and subscripts on hardcopy terminals. If a hardcopy terminal can eject to the next page (form feed), specify the as \texttt{ff} (usually Ctrl-L) capability.

If there is a command to repeat a given character a given number of times (to save time transmitting a large number of identical characters), this can be indicated with the \texttt{rep} parameterized string. The first parameter is the character to be repeated, and the second is the number of times to repeat it. Thus following:

\begin{verbatim}
 tpars(repeat_char,'x',10)
\end{verbatim}

is the same as

\begin{verbatim}
 xxxxxxxxxx
\end{verbatim}

If the terminal has a settable command character, such as the Tektronix 4025, indicate this with the \texttt{cmdch} capability. A prototype command character is chosen that is used in all capabilities. This character is given in the \texttt{cmdch} capability to identify it. On some UNIX systems, if the \texttt{CC} environment variable exists, all occurrences of the prototype character are replaced with the character in the \texttt{CC} variable.

Terminal descriptions that do not represent a specific kind of known terminal such as switch, dialup, patch, and network, should include the \texttt{gn} (generic) capability. This capability allows programs to return errors if they cannot talk to the terminal. The \texttt{gn} capability does not apply to virtual terminal descriptions for which the escape sequences are known. If a terminal is supported by the UNIX system virtual terminal protocol, use the \texttt{vt} capability to define its terminal number.

If a terminal uses xon/xoff handshaking for the flow control, its description should include the \texttt{xon} capability. You should still include padding information as well so that routines can make better decisions about costs. However, actual pad characters are not transmitted. To specify sequences to turn on and off
xon/xoff handshaking, use the *smxon* and *rmxon* capabilities. If the characters used for handshaking are not ^S and ^Q, use the *xonc* and *xoffc* capabilities to define them.

If a terminal has a meta key that acts as a shift key to set the eighth bit of any character transmitted, identify the key with the *km* capability. Otherwise, software assumes that the eighth bit is parity, and it will usually be cleared. If strings exist to turn this meta mode on and off, they can be given as the *smm* and *rmm* capabilities.

If a terminal has more lines of memory than fit on the screen at once, use the *lm* capability to define the number of lines of memory. A value of *lm#0* indicates that the number of lines is not fixed, but that there are still more lines of memory than fit on the screen.

Media copy strings that control an auxiliary printer connected to the terminal are identified with the following capabilities:

- **mc0** Prints the contents of the screen
- **mc4** Turns off the printer, and
- **mc5** Turns on the printer. When the printer is on, all text sent to the terminal is sent to the printer. It is undefined whether the text is also displayed on the terminal screen when the printer is on.
- **mc5p** Leaves the printer on for a specified number of characters and then turns the printer off. The parameter passed to *mc5p* should not exceed 255.

If the terminal screen does not display the text when the printer is on, specify the *mc5i* capability to signify a silent printer. All text, including the *mc4*, is transparently passed to the printer while an *mc5p* is in effect.

### Status Lines

You can use the *terminfo* entry to indicate that the terminal has an extra status line that is not normally used by software. If the status line is viewed as an extra line below the bottom line, into which the cursor can be addressed normally, the *hs* capability should be given. Special strings to go to the beginning of the status line and to return from the status line can be given as the *tsl* and *fsl* capabilities, respectively. (The *fsl* must leave the cursor position in the same place it was before the *tsl*. If necessary, the *sc* string and the *rc* string can be included in *tsl* and *fsl* to get this effect.) The *tsl* capability takes one parameter, which is the column number of the status line to which the cursor is to be moved.

If escape sequences and other special commands, such as tab, work while in the status line, specify the *eslok* capability. A string that turns off the status line (or otherwise erases its contents) should be given as *dsl*. If the terminal has commands to save and restore the position of the cursor, give them as *sc* and *rc* capabilities. The status line is normally assumed to be the same width as the rest of the screen, such as *cols*. If the status line is a different width (possibly because the terminal does not allow an entire line to be loaded), the width, in columns, can be indicated with the *wsl* numeric parameter.

### Line Graphics

If the terminal has a line drawing alternate character set, specify the mapping of glyph to character in the *acsc* capability. The definition of this string is based on the alternate character set used in the DEC VT100 terminal, extended slightly with some characters from the AT&T4410v1 terminal. Use the following to define the string:

<table>
<thead>
<tr>
<th>Glyph Name</th>
<th>vt100+ Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>arrow pointing right</td>
<td>+</td>
</tr>
<tr>
<td>arrow pointing left</td>
<td>,</td>
</tr>
<tr>
<td>arrow pointing down</td>
<td>.</td>
</tr>
<tr>
<td>solid square block</td>
<td>0</td>
</tr>
<tr>
<td>lantern symbol</td>
<td>l</td>
</tr>
<tr>
<td>arrow pointing up</td>
<td>-</td>
</tr>
</tbody>
</table>
The best way to describe a new terminal's line graphics set is to add a third column to the above table with the characters for the new terminal that would produce the appropriate glyph when the terminal is in alternate character set mode. For example:

glyph name | vt100+ character | tty character
---|---|---
upper left corner | l | R
lower left corner | m | F
upper right corner | k | T
lower right corner | j | G
horizontal line | q | ,
vertical line | x | .

Then, you specify the acsc capability by specifying the characters from left to right as follows:
acsc=lRmFkTjGq\,x.

**Color Manipulation**

There are two methods of color manipulation, the HP method and the Tektronix method. Most existing color terminals belong to one of these two classes. The Tektronix method uses a set of \( N \) predefined colors (usually 8) from which a user can select *current* foreground and background colors. Thus, the terminal can support up to \( N \) colors mixed into \( N \times N \) color-pairs that are displayed on the screen at the same time.

The HP method restricts the user from both defining the foreground independently of the background or the background independently of the foreground. Instead, the user must define an entire color-pair at once. Up to \( M \) color-pairs, made from \( 2^M \) different colors, can be defined this way.

The numeric variables *colors* and *pairs* define the number of colors and color-pairs that the terminal can display on the screen at one time. If a terminal can change the definition of a color, you should specify the ccc capability. To change the definition of a color using the Tektronix method, use the initc capability. This capability requires four parameters: a color number ranging from 0 to colors-1 and three Red, Green, Blue (RGB) values ranging from 0 to 1,000.
Tektronix 4100 series terminals use a type of color notation called HLS (Hue Lightness Saturation) instead of RGB color notation. For such terminals, you should define the hls boolean capability. The last three arguments to the initc capability would then be HLS values where H ranges from 0 to 360 and L and S range from 0 to 100.

**Note:** If a terminal can change the definitions of colors but uses a color notation different from RGB or HLS, you must develop a mapping to either RGB or HLS.

To set current foreground and background to a given color, use the setf and setb capabilities. These capabilities require a single parameter that specifies the number of the color. To use the HP method to initialize a color-pair, use the initp capability. This capability requires seven parameters:
- the number of the color-pair in the range of 0 to pairs -1
- three RGB values for the foreground
- three RGB values for the background

When you use the initc or initp capabilities, be sure you specify the values in the order red, green, blue or hue, lightness, saturation, respectively. To make a color-pair current, use the scp capability. This capability takes one parameter, the number of the color-pair.

Some terminals erase areas of the screen with the current background color. In such cases, define the bce capability. The op capability contains a sequence for setting the foreground and the background colors to what they were at the terminal start-up time. Similarly, the oc capability contains a control sequence for setting all colors or -pairs to the values they had at the terminal start-up time.

Some color terminals substitute color for video attributes. Such video attributes should not be combined with colors. You should pack information about these video attributes into the ncv capability. There is a one-to-one correspondence between the nine least significant bits of that variable and the video attributes. The following table depicts this correspondence:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>NCV Bit Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>A_STANDOUT</td>
<td>0</td>
</tr>
<tr>
<td>A_UNDERLINE</td>
<td>1</td>
</tr>
<tr>
<td>A_REVERSE</td>
<td>2</td>
</tr>
<tr>
<td>A_BLINK</td>
<td>3</td>
</tr>
<tr>
<td>A_DIM</td>
<td>4</td>
</tr>
<tr>
<td>A_BOLD</td>
<td>5</td>
</tr>
<tr>
<td>A_INVIS</td>
<td>6</td>
</tr>
<tr>
<td>A_PROTECT</td>
<td>7</td>
</tr>
<tr>
<td>A_ALTCHARSET</td>
<td>8</td>
</tr>
</tbody>
</table>

When a particular video attribute should not be used with colors, the corresponding ncv bit should be set to 1. Otherwise, set the bit to 0. For example, if the terminal uses colors to simulate reverse video and bold, bits 2 and 5 should be set to 1. The resulting values for ncv will be 22.

**Special Cases**

Some terminals require special support by the terminfo database. These terminals are not deficient. These terminals have hardware that may be slightly different than what the terminfo database expects of most terminals. Some of the special cases are discussed in this section. The programmer’s manual for a terminal should provided all the information you need to code a terminfo description for the terminal.

For terminals that do not allow the ~ (tilde) character, use the hz capability.
Descriptions of terminals that ignore a line-feed character immediately after an \texttt{am} wrap should include the \texttt{xenl} capability. Those terminals whose cursor remains on the right-most column until another character is received rather than wrapping immediately upon receiving the right-most character, should also use the \texttt{xenl} capability.

If \texttt{el} capability is required to get rid of standout (instead of merely writing normal text on top of it), then you should specify \texttt{xhp} capability.

Terminals for which tabs change all moved characters into blanks should indicate the \texttt{xt} capability (destructive tabs). This capability is interpreted to mean that it is not possible to position the cursor on top of the pads inserted for standout mode. Instead, it is necessary to erase standout mode using delete and insert line.

A terminal that is unable to correctly transmit the ESC (escape) or Ctrl-C characters should specify the \texttt{xsb} capability, indicating that the F1 key is used for ESC and the F2 key is used for Ctrl-C.

Other specific terminal problems can be corrected by adding more capabilities.

**Similar Terminals**

If two terminals are very similar, you can define one as being just like the other with the \texttt{use} string capability. You can also use all of the definitions from an existing description and identify exceptions. The capabilities given before the \texttt{use} capability override those in the terminal type called by the \texttt{use} capability. To cancel a capability place \texttt{xx@} to the left of the \texttt{use} capability definition, where \texttt{xx} is the capability. For example, the entry:

\begin{verbatim}
term-nl | Terminal smkx@, rmkx@, use=term
\end{verbatim}

defines a terminal that does not have either the \texttt{smkx} or the \texttt{rmkx} capability, and hence does not turn on the function key labels when in visual mode. This is useful for different terminal modes or for different user preferences. You can specify more than one \texttt{use} capability.

**Printer Capabilities**

The \texttt{terminfo} database allows you to define the capabilities of printers as well as terminals. To find out what capabilities are available for printers as well as for terminals, see the two lists under Terminal Capabilities that the list the capabilities by variable and by capability name.

**Rounding Values**

Because parameterized string capabilities work only with integer values, we recommend that \texttt{terminfo} designers create strings that expect rounded numeric values. Programmers should always round values to the nearest integer before using them with a parameterized string capability.

**Printer Resolution**

A printer's resolution is the smallest spacing of characters it can achieve. In general, printers have independent resolution horizontally and vertically. To determine the vertical resolution of a printer, measure the smallest achievable distance between consecutive printing baselines. To determine the horizontal resolution, measure the smallest achievable distance between the left-most edges of consecutive printed, identical, characters.

The \texttt{terminfo} database assumes all printers are capable of printing with a uniform horizontal and vertical resolution. The \texttt{terminfo} database currently interacts with printers as if they print inside a uniform matrix. All characters are printed at fixed positions relative to each cell in the matrix. Furthermore, each cell has the same size given by the smallest horizontal and vertical step sizes dictated by the resolution.

Many printers are capable of proportional printing where the horizontal spacing depends on the size of the last character printed. The \texttt{terminfo} database does not make use of this capability, although it does provide enough capability definitions to allow an application to simulate proportional printing.
A printer must not only be able to print characters as close together as the horizontal and vertical 
resolutions suggest, but also of moving to a position that is an integral multiple of the smallest distance 
away from a previous position. Thus, printed characters can be spaced apart a distance that is an integral 
multiple of the smallest distance, up to the length of width of a single page.

Some printers can have different resolutions depending on different modes. In normal mode, the existing 
terminfo capabilities are assumed to work on columns and lines, just like a video terminal. For example, 
the old lines capability specify the length of a page in lines, and the cols capability specifies the width of a 
page in columns. In micro mode many terminfo capabilities work on increments of lines and columns. 
With some printers, the micro mode may exist concurrently with normal mode, so that all the capabilities 
work at the same time.

**Specifying Printer Resolution**
You can specify a printer's printing resolution with several different capabilities. Each capability specifies 
distance in a different way. The following capabilities define print resolution:

<table>
<thead>
<tr>
<th>Capability</th>
<th>Defined as</th>
</tr>
</thead>
<tbody>
<tr>
<td>orhi</td>
<td>steps per inch horizontally</td>
</tr>
<tr>
<td>orvi</td>
<td>steps per inch vertically</td>
</tr>
<tr>
<td>orc</td>
<td>steps per column</td>
</tr>
<tr>
<td>orl</td>
<td>steps per line</td>
</tr>
</tbody>
</table>

When printing in normal mode, each character printed causes the printer to move to the next column, 
except in special cases described later. The distance moved is the same as the per-column resolution. 
Some printers cause an automatic movement to the next line when a character is printed in the rightmost 
position. The vertical distance moved is the same as the per-line resolution. When printing in micro mode, 
these distances can be different, and may be zero for some printers. The following specify printer 
resolution automatic motion after printing:

<table>
<thead>
<tr>
<th>Capability</th>
<th>Defined as</th>
</tr>
</thead>
<tbody>
<tr>
<td>orc</td>
<td>Steps moved horizontally in normal mode.</td>
</tr>
<tr>
<td>orl</td>
<td>Steps moved vertically in normal mode.</td>
</tr>
<tr>
<td>mcs</td>
<td>Steps moved horizontally in micro mode.</td>
</tr>
<tr>
<td>mls</td>
<td>Steps moved vertically in micro mode.</td>
</tr>
</tbody>
</table>

Some printers can print wide characters. The distance moved when a wide character is printed in normal 
mode may be different from when a regular width character is printed. The distance moved when a wide 
character is printed in micro mode may also be different from when a regular character is printed in micro 
mode, but the differences are assumed to be related.

If the distance moved for a regular character is the same in normal mode or micro mode (mcs=ocs), then 
the distance moved for a wide character is also the same in both modes. This does not mean the normal 
character distance is necessarily the same as the wide character distance, just that the distances do not 
change with a change from normal to micro mode. Use the widcs capability to specify the printer 
resolution when the automatic motion after printing a wide character is the same in both normal or micro 
mode.

If the distance moved for a regular character is different in micro mode from the distance moved in normal 
mode (mcs<orc), you can assume the micro mode distance is the same for a wide character printed in 
micro mode. In this case, you use the mcs capability to specify the distance moved. The printer uses the 
value you specify for both regular and wide characters

A printer may use control sequences to change the number of columns per inch (the character pitch) and 
to change the number of lines per inch (the line pitch). If these are used, the resolution of the printer 
changes but the type of change depends on the printer.
<table>
<thead>
<tr>
<th>Capability</th>
<th>Defined as</th>
</tr>
</thead>
<tbody>
<tr>
<td>cpi</td>
<td>Change character pitch.</td>
</tr>
<tr>
<td>cpix</td>
<td>If set, cpi changes orhi, otherwise the cpi capability changes the orc value.</td>
</tr>
<tr>
<td>lpi</td>
<td>Change line pitch</td>
</tr>
<tr>
<td>lpix</td>
<td>If set, lpi changes the orvi value, otherwise the orl value is changed.</td>
</tr>
<tr>
<td>chr</td>
<td>Changes steps per column.</td>
</tr>
<tr>
<td>cvr</td>
<td>Changes steps per line.</td>
</tr>
</tbody>
</table>

The cpi and lpi string capabilities have a single argument, the pitch in columns (or characters) and lines per inch, respectively. The chr capability and cvr string capabilities each have a single argument, the number of steps per column and line, respectively.

Using any of the control sequences in these strings implies a change in some of the values of the orc, orhi, orl, and orvi capabilities. Also, the distance moved when a wide character is printed, specified by the widcs capability, changes in relation to the orc value. The distance moved when a character is printed in micro mode, mcs, changes similarly, with one exception: if the distance is 0 or 1, then no change is assumed.

Programs that use the cpi, lpi, chr, or cvr capability should recalculate the printer resolution and should recalculate other values. For more information, see [Effect of Changing Printing Resolution](#).

The following figure, "Specification of Printer Resolution Effects of Changing the Character/Line Pitches" shows the effects on printer resolution before and after a change.
Vcpi, Vlpi, Vchr, and Vcvr are the arguments used with cpi, lpi, chr, and cvr respectively. The dagger symbol indicates the old value.

Capabilities that Cause Movement
In the following descriptions, movement refers to the motion of the current position. With video terminals this would be the cursor; with some printers this is the carriage position. Other printers have different equivalents. In general, the current position is where a character would be displayed if printed.

The terminfo database has string capabilities for control sequences that cause movement a number of full columns or lines. It also has equivalent string capabilities for control sequences that cause movement a number of small steps. The following are the string capabilities for motion:

<table>
<thead>
<tr>
<th>Capability</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mcub1</td>
<td>Move 1 step left.</td>
</tr>
<tr>
<td>mcuf1</td>
<td>Move 1 step right.</td>
</tr>
<tr>
<td>mcu1</td>
<td>Move 1 step up.</td>
</tr>
<tr>
<td>mcud1</td>
<td>Move 1 step down.</td>
</tr>
<tr>
<td>mcub</td>
<td>Move N steps left.</td>
</tr>
</tbody>
</table>

Figure 1. This illustration shows the effects of changing character pitch and line pitch on printer resolution.
### Capability Description

- **mcuf**
  - Move $N$ steps right.

- **mcuu**
  - Move $N$ steps up.

- **mcud**
  - Move $N$ steps down.

- **mhpa**
  - Move $N$ steps from the left.

- **mvpa**
  - Move $N$ steps from the top.

The last six strings are each used with a single $N$ argument.

Sometimes the motion is limited to less than the width or length of a page. Also, some printers do not accept absolute motion to the left of the current position. The following capabilities limit motion:

<table>
<thead>
<tr>
<th>Capability</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mjump</td>
<td>Limits the use of mcub1, mcuf1, mceu1, and mcud1 capabilities.</td>
</tr>
<tr>
<td>maddr</td>
<td>Limits the use of the mhpa and mvpa capabilities.</td>
</tr>
<tr>
<td>xhpa</td>
<td>If set, the hpa and mhpa capabilities are negated.</td>
</tr>
<tr>
<td>xvpa</td>
<td>If set, the vpa and mvpa capabilities are negated.</td>
</tr>
</tbody>
</table>

If a printer needs to be in *micro mode* for the motion capabilities to work, you can define a string capability to contain the control sequence to enter and exit micro mode. A boolean is available for those printers where using a carriage return causes an automatic return to normal mode. The following capabilities are related to micro mode behavior:

<table>
<thead>
<tr>
<th>Capability</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>smicm</td>
<td>Enter micro mode.</td>
</tr>
<tr>
<td>rmicm</td>
<td>Exit micro mode.</td>
</tr>
<tr>
<td>crxm</td>
<td>Using the key specified by the cr capability exits micro mode.</td>
</tr>
</tbody>
</table>

The movement made when a character is printed in the rightmost position varies among printers. Some make no movement, some move to the beginning of the next line, others move to the beginning of the same line. The **terminfo** database has boolean capabilities that description all three cases. The sam capability specifies that the printer automatically moves to the beginning of the same line after the character is printed in the rightmost margin.

Some printers can be put in a mode where the normal direction of motion is reversed. This mode is especially useful when there exists no capabilities for leftward or upward motion, you can build these capabilities from the motion reversal capability and the rightward or downward motion capabilities. It is best to leave it up to an application to build the leftward or upward capabilities, though, and not enter them into to the **terminfo** database. This allows several reverse motions to be strung together without intervening wasted steps that leave and reenter reverse mode. The following capabilities control entering and exiting reverse modes:

<table>
<thead>
<tr>
<th>Capability</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>slm</td>
<td>Reverse sense of horizontal motions.</td>
</tr>
<tr>
<td>rlm</td>
<td>Restore sense of horizontal motions.</td>
</tr>
<tr>
<td>sum</td>
<td>Reverse sense of vertical motions.</td>
</tr>
<tr>
<td>rum</td>
<td>Restore sense of vertical motions.</td>
</tr>
</tbody>
</table>

The following capabilities affect the screen while the horizontal motions are reversed:

<table>
<thead>
<tr>
<th>Capability</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mcub1</td>
<td>Move 1 step right.</td>
</tr>
<tr>
<td>mcuf1</td>
<td>Move 1 step left.</td>
</tr>
</tbody>
</table>
Capability | Description
---|---
mcul | Move \( N \) steps right.
mcufl | Move \( N \) steps left.
cubl | Move 1 column right.
cufl | Move 1 column left.
cub | Move \( N \) columns right.
cufl | Move \( N \) columns left.

The following capabilities affect the screen while the vertical motions are reversed:

| Capability | Description |
---|---|
mcuu | Move 1 step down. |
mucud | Move 1 step up. |
mcuu | Move \( N \) steps down. |
mucud | Move \( N \) steps up. |
cuu | Move 1 line down. |
cuud | Move 1 line up |
cuu | Move \( N \) lines down. |
cud | Move \( N \) lines up. |

The reverse motion mode should not affect the mypress and mhp_press absolute motion capabilities. The reverse vertical motion mode should, however, also reverse the action of the line wrapping that occurs when a character is printed in the right-most position. Thus printers that have the standard terminfo capability am defined should move to the beginning of the previous line when a character is printed on the right-most position and the printer is in reverse-vertical motion mode.

The action when any other motion capabilities are used in reverse motion modes is not defined. Thus, programs must exit reverse motion modes before using other motion capabilities.

Two miscellaneous capabilities complete the list of new motion capabilities, the docr and the zeron capability. The docr capability provides a list of control characters that cause a carriage return. This capability is useful for printers that move the current position to the beginning of a line when certain control characters, like line-feed or form-feed are used. The zeron capability prevents automatic motion after printing a single character. This capability suspends the motion that normally occurs after printing a character.

### Margins

The terminfo database provides two strings for setting margins on terminals: one for the left and one for the right margin. Printers, however, have two additional margins for the top and bottom margins of each page. Furthermore, some printers do not require using motion strings to move the current position to a margin and fixing the margin there, as with existing capabilities, but require the specification of where a margin should be regardless of the current position. Therefore, the terminfo database offers six additional strings for defining margins with printers. The following capabilities affect margins:

| Capability | Definition |
---|---|
smgl | Set left margin at the current column. |
smgr | Set right margin at the current column. |
smgb | Set the soft bottom margin at the current line. |
smgt | Set the soft top margin at the current line. |
smgbp | Set the soft bottom margin at line \( N \). |
smglp | Set the soft left margin at column \( N \). |
smgrp | Set the soft right margin at column \( N \). |
smgrp | Set soft top margin at line \( N \). |
The last four strings are used with a single $N$ parameter. This parameter specifies a line or column number, where 0 is the top line and column 0 is the left-most column.

**Note:** Not all printers use 0 for the top line or the left-most column.

All margins can be cleared with the `mgc` capability.

## Shadows, Italics, Wide Characters, Superscripts, and Subscripts

Five new sets of strings are used to describe the capabilities that printers have of enhancing printed text. The following define enhanced printing capabilities:

<table>
<thead>
<tr>
<th>Capability</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>sshm</td>
<td>Enter shadow-printing mode.</td>
</tr>
<tr>
<td>rshm</td>
<td>Exit shadow-printing mode.</td>
</tr>
<tr>
<td>sitm</td>
<td>Enter italicizing mode.</td>
</tr>
<tr>
<td>ritm</td>
<td>Exit italicizing mode.</td>
</tr>
<tr>
<td>swidm</td>
<td>Enter wide-character mode.</td>
</tr>
<tr>
<td>rwidm</td>
<td>Exit wide-character mode.</td>
</tr>
<tr>
<td>ssupm</td>
<td>Enter superscript mode.</td>
</tr>
<tr>
<td>rsupm</td>
<td>Exit superscript mode.</td>
</tr>
<tr>
<td>supcs</td>
<td>List of characters available as superscripts.</td>
</tr>
<tr>
<td>ssubm</td>
<td>Enter subscript mode.</td>
</tr>
<tr>
<td>rsubm</td>
<td>Exit subscript mode.</td>
</tr>
<tr>
<td>subcs</td>
<td>List of characters available as subscripts.</td>
</tr>
</tbody>
</table>

If a printer requires the `sshm` control sequence before every character to be shadow-printed, the `rshm` string is left blank. Thus, programs that find a control sequence in `sshm` but none in shadow printing mode should use the control sequence specified by the `sshm` capability before every character to be shadow printed. Otherwise, the control sequence should be used once before the set of characters to be shadow-printed, followed by exiting shadow-printing mode.

The `terminfo` database also has a capability for printing emboldened text, the `bold` capability. While shadow printing and emboldened printing are similar in that they darken the text, many printers produce these two types of print in slightly different ways. Generally emboldened printing is done by overstriking the same character one or more times. Shadow printing likewise usually involves overstriking, but with a slight movement up and/or to the side so that the character is fatter.

It is assumed that enhanced printing modes are independent modes, so that it would be possible, for instance, to shadow print italicized subscripts.

As mentioned earlier, the amount of motion automatically made after printing a wide character should be given in the `widcs` capability.

If only a subset of the printable ASCII characters can be printed as superscripts or subscripts, they should be listed in the `supcs` or `subcs` capabilities, respectively. If the `ssupm` or `ssubm` strings contain control sequences, but the corresponding `supcs` or `subcs` strings are empty, it is assumed that all printable ASCII characters are available as superscripts or subscripts.

Automatic motion made after printing a superscript or subscript is assumed to be the same as for regular characters. For example, printing any of the following result in equivalent motion:

```
Bi 
B i
```

The boolean capability `msgr` describes whether an application can use motion control sequences while in standout mode. This capability is extended to cover the enhanced printing modes added here. The `msgr` capability should be set for those printers that accept any motion control sequences without affecting
shadow, italicized, widened, superscript, or subscript printing. Conversely, if the mgsr capability is not set, a program should end these modes before attempting any motion.

**Alternate Character Sets**

In addition to allowing you to define line graphics, the *terminfo* database also lets you define alternate character sets. The following capabilities cover printers and terminals with multiple selectable or definable character sets:

<table>
<thead>
<tr>
<th>Capability</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>scs</td>
<td>Select character set N. The N parameter specifies a number from 0 to 63 that identifies a character set.</td>
</tr>
<tr>
<td>scsd</td>
<td>Start definition of character set N, M characters. The N parameter specifies a number from 0 to 63 that identifies a character set and the M parameter specifies the number of characters in the set.</td>
</tr>
<tr>
<td>defc</td>
<td>Defines a character A to be B dots wide with a descender D. The A parameter is the ASCII code representation for the character. The B parameter specifies the width of the character in dots. The D parameter specifies whether the character is a descender or not. If the character is a descender, specify a 1 for the D parameter. Otherwise, specify a 1. This string is followed by a string of image-data bytes that describe how the character looks.</td>
</tr>
<tr>
<td>rcsd</td>
<td>End definition of character set N. The N parameter specifies a number from 0 to 63 that identifies a character set.</td>
</tr>
<tr>
<td>csnm</td>
<td>List of character set names.</td>
</tr>
<tr>
<td>daisy</td>
<td>Indicates the printer has manually changed print-wheels.</td>
</tr>
</tbody>
</table>

Character set 0 is the default character set. This is the set that is present after the printer is initialized. Not every printer supports 64 character sets. If you specify a set that a printer does not support, the *tparm* subroutine returns a null result.

If your application must define a character before using it, use the *scsd* control sequence before defining the character set, and the *rcsd* after. If you specify an invalid character set for either of these capabilities, the *tparm* subroutine returns a null resolution. If your application must select a character set after it is defined, the *scs* control sequence should follow the *rcsd* control sequence. By examining the results of using each of the *scs*, *scsd*, and *rcsd* strings with a character set number in a call to the *tparm* subroutine, a program can determine which of the three are needed.

Between use of the *scsd* and *rcsd* strings, the *defc* string should be used to define each character. To print any character on printers defined in the *terminfo* database, the ASCII cod is sent to the printer. This is true for characters in an alternate set as well as normal characters. Thus, the definition of a character includes the ASCII code that represents it. In addition, the width of the character includes the ASCII code that represents it. In addition, the width of the character in dots is given, along with an indication of whether the character is a descender. A descender is a character whose shape extends below the baseline, for example the character g is a descender. The width of the character is dots also indicates the number of image-data bytes that will follow the *defc* string. These image-data bytes indicate where in a dot-matrix pattern ink should be applied to draw the character. The number of these bytes and their form are defined below under Dot-Mapped Graphics.

It is easiest for the creator of *terminfo* entries to refer to each character set by number. However, these numbers will be meaningless to the application developer. The *csnm* capability alleviates this problem by providing names for each number.

When used with a character set number in a call to the *tparm* subroutine, the *csnm* capability produces the equivalent name. Use these names as a references only. No naming convention is implied, although anyone who creates a *terminfo* entry for a printer should use names consistent with the names found in user documents for the printer. Application developers should allow a user to specify a character set by
number (leaving it up to the user to examine the csnm string to determine the correct number), or by name, where the application examines the csnm capability to determine the corresponding character set number.

The alternate character set capabilities are likely to be used only with dot-matrix printers. If they are not available, do not define these strings. For printers that have manually changed print-wheels or font cartridges, set the boolean daisy capability.

**Dot-Matrix Graphics**

Dot-matrix printers typically have the capability to reproduce raster-graphics images. Three new numeric capabilities and three new string capabilities can help a program draw raster-graphic images independent of the type of dot-matrix printer or the number of pins or dots the printer can handle at one time. The dot-matrix capabilities are as follows:

<table>
<thead>
<tr>
<th>Capability</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>npins</td>
<td>Number of pins $N$ in the print-head. The $N$ parameter specifies the number of pins.</td>
</tr>
<tr>
<td>spinv</td>
<td>Spacing of pins vertically in pins per inch.</td>
</tr>
<tr>
<td>spinh</td>
<td>Spacing of dots horizontally in dots per inch.</td>
</tr>
<tr>
<td>porder</td>
<td>Matches software bits to print-head pins.</td>
</tr>
<tr>
<td>sbim</td>
<td>Start printing bit image graphics, $B$ bits wide. The $B$ value specifies the width of the image in dots.</td>
</tr>
<tr>
<td>rbim</td>
<td>End printing bit image graphics.</td>
</tr>
</tbody>
</table>

The model of dot-matrix or raster-graphics that the terminfo database presents is similar to the technique used for most dot-matrix printers. Each pass of the printer’s print-head is assumed to produce a dot-matrix that is $N$ dots high and $B$ dots wide. This is typically a wide, squat, rectangle of dots. The height of this rectangle in dots varies from one printer to the next. This is given in the npins numeric capability. The size of the rectangle in fractions of an inch will also vary. The size can be deduced from the spinv and spinh numeric capabilities. With these three values an application can divide a complete raster-graphics image into several horizontal strips, perhaps interpolating to account for different dot spacing vertically and horizontally.

The sbim and rbim capabilities are used to start and end a dot-matrix image, respectively. The sbim capability is used with a single argument that gives the width of the dot-matrix in dots. A sequence of image-data bytes are sent to the printer after the sbim capability and before the rbim string. The number of bytes is an integral multiple of the width of the dot-matrix. The multiple and the form of each byte is determined by the porder capability is described below.

The porder capability is a comma-separated list of pin numbers. The position of each pin number in the list corresponds to a bit in a data byte. The pins are numbered consecutively from 1 to npins, with 1 being the top pin. The term pin is used loosely here. Ink-jet dot matrix printers don’t have pins but they do have an equivalent method of applying a single dot of ink to paper. The bit positions in porder are in groups of 8, with the first position in each group the most significant bit and the last position the least significant bit.

The image-data bytes are computed from the dot-matrix image, mapping vertical dot positions in each print-head pass into eight-bit bytes, using a 1 bit where ink should be applied and 0 where no ink should be applied. If a position is skipped in porder, a 0 bit is used. There must be a multiple of 8 bit positions used or skipped in porder. If not, 0 bits are used to fill the last byte in the least significant bits.

**Effect of Changing Printing Resolution**

If the control sequences to change the character pitch or the line pitch are used, the pin or dot spacing may change. The following capabilities change pitch on dot-matrix graphics:

<table>
<thead>
<tr>
<th>Capabilities</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>cpi</td>
<td>Change the character pitch.</td>
</tr>
<tr>
<td>cpix</td>
<td>If set, cpi changes spinh.</td>
</tr>
</tbody>
</table>
### Capabilities

<table>
<thead>
<tr>
<th>Capability</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>lpi</strong></td>
<td>Change line pitch.</td>
</tr>
<tr>
<td><strong>lpix</strong></td>
<td>If set, lpi changes <strong>spinv</strong>.</td>
</tr>
</tbody>
</table>

Programs that use **cpi** or **lpi** should recalculate the dot spacing. The following figure “Dot-Matrix Graphics Effects of Changing the Character/Line Pitches” shows graphics both before and after a change in pitch.

### Dot-Matrix Graphics Effects of Changing the Character/Line Pitches

<table>
<thead>
<tr>
<th>Using <strong>cpi</strong> with <strong>cpi</strong> clear:</th>
<th>Using <strong>cpi</strong> with <strong>cpi</strong> set:</th>
</tr>
</thead>
<tbody>
<tr>
<td>spinh `'</td>
<td>spinh=spinh' '. orhi'</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Using <strong>lpi</strong> with <strong>lpi</strong> clear:</th>
<th>Using <strong>lpi</strong> with <strong>lpi</strong> set:</th>
</tr>
</thead>
<tbody>
<tr>
<td>spinv `'</td>
<td>spinv=spinv' '. orhi'</td>
</tr>
</tbody>
</table>

| Using **chr:**                  |                                  |
| spinh `'                        | spinh                            |

| Using **cvr:**                  |                                  |
| spinv `'                        | spinv                            |

*Figure 2. This illustration shows the effects of changing character pitch and line pitch on dot-matrix graphics.*

The **orhi'** and **orhi** values are the values of the horizontal resolution in steps per inch, before using **cpi** and after using **cpi**, respectively. Likewise, **orvi'** and **orvi** are the values of the vertical resolution in steps per inch, before using **lpi** and after using **lpi**, respectively. Thus, the changes in the dots per inch for dot-matrix graphics follow the changes in steps per inch for printer resolution.

### Print Quality

Many dot-matrix printers can alter the dot spacing of printed text to produce near letter-quality printing or draft-quality printing. Usually, it is important to be able to choose one or the other because the rate of printing generally falls off as the quality improves. The capabilities that specify print quality are the following:

<table>
<thead>
<tr>
<th>Capability</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>snlq</strong></td>
<td>Set near-letter quality print.</td>
</tr>
<tr>
<td><strong>snrq</strong></td>
<td>Set normal quality print.</td>
</tr>
<tr>
<td><strong>sdrfq</strong></td>
<td>Set draft-quality print.</td>
</tr>
</tbody>
</table>

The capabilities are listed in decreasing levels of quality. If a printer does not have all three levels, one or two of the strings should be left blank as appropriate.
Printing Rate and Buffer Size
Because there is no standard protocol that synchronizes a printer with a program, and because modern printers can buffer data before printing it, a program generally cannot determine at any time what has printed. Two new numeric capabilities can help a program estimate what has printed, the \texttt{cps} and \texttt{bufsz} capabilities.

The \texttt{cps} capability specifies the nominal print rate in characters per second. The \texttt{cps} capability is the nominal or average rate at which the printer prints characters. If this value is not given, estimate the rate at one-tenth the prevailing baud rate.

The \texttt{bufsz} capability defines a terminal’s buffer capacity in characters. The \texttt{bufsz} value is the maximum number of subsequent characters buffered before the guaranteed printing of an earlier character, assuming proper flow control was used. If this value is not given it is assumed that the printer does not buffer characters, but prints them as they are received.

As an example, if a printer has a 1000-character buffer, then sending the letter “a” followed by 1000 additional characters is guaranteed to cause the letter “a” to print. If the same printer prints at the rate of 100 characters per second, then it should take 10 seconds to print all the characters in the buffer, less if the buffer is not full. By keeping track of the characters sent to a printer, and knowing the print rate and buffer size, a program can synchronize itself with the printer.

Most printer manufacturers advertise the maximum print rate, not the nominal print rate. A good way to get a value to put in for \texttt{cps} is to generate a few pages of text, count the number of printable characters, then see how long it takes to print the text.

Applications that use these values should recognize the variability in the print rate. Straight text, in short lines, with no embedded control sequences will probably print at close to the advertised print rate and probably faster than the rate in \texttt{cps}. Graphics data with a lot of control sequences, or very long lines of text, will print at well below the advertised rate and below the rate in \texttt{cps}. If the application is using \texttt{cps} to decide how long it should take a printer to print a block of text, the application should pad the estimate. If the application is using \texttt{cps} to decide how much text has already been printed, it should shrink the estimate. The application errs in favor of the user, who wants, above all, to see all the output in its correct place.

Database File Names
Compiled \texttt{terminfo} file descriptions are placed in subdirectories under the \texttt{/usr/share/lib/terminfo} directory to avoid performing linear searches through a single directory containing all of the \texttt{terminfo} file description files. A given description file is stored in the \texttt{/usr/share/lib/terminfo/c/name} file, where \texttt{name} is the name of the terminal, and \texttt{c} is the first letter of the terminal name. For example, the compiled description for the terminal \texttt{term4-nl} can be found in the file \texttt{/usr/share/lib/terminfo/t/term4-nl}. You can create synonyms for the same terminal by making multiple links to the same compiled file. (See the \texttt{ln} command on how to create multiple links to a file.)

Example
The following \texttt{terminfo} entry describes a terminal:

\begin{verbatim}
ht|High Function Terminal,  
  cr=^M, cud1=\E[B, ind=\E[S, bel=^G, ill=\E[L, am, 
  cub1=H, ed=\E[J, e1=\E[K, clear=\E[H\E[J, 
  cup=\E[ip1d;ip2dH, cols=80, lines=25, 
  dch1=\E[P, d1=\E[0, home=\E[H, 
  ich=\E[2;1d0, ich1=\E[0, smir=\E[6, rmir=\E[6, 
  bold=\E[m, rev=\E[7m, blink=\E[5m, invis=\E[8m, sgr0=\E[0m, 
  sgr=\E[?2p1t7;?2p2t4;?2p3t7;?2p4t5;?2p6t1;m, 
  kcu1=\E[A, kcu1=\E[B, kcu1=\E[D, 
  kcu1=\E[C, khome=\E[H, kbs=^H,
\end{verbatim}
The following terminfo entry describes a terminal:

```
ibm3161|ibm3163|wyse60-316X|wyse60-316X|IBM 3161/3163 display,
    am, mir, cr=M, ind=^J,
    cols=80, it=^?, lines=24,
    kich1=EP\040\010, ed=EO, el=EI, cup=\E%p1%' '=%c%p2%'
    '=%c,
    clear=\EH\040, dch1=EQ, dl1=EO, cud1=EB,
    cub1=ED, blink=\E4D, bol=\E4H,
    sgr0=\E40, ed=\E40, rev=\E4A, cu1=\E4A,
    rmso=\E4A, smul=\E4A, rmul=\E4A, cu1=\E4A,
    smcup=\E4A, rmcup=\E4A, smgr,
    home=\EH, bel=^G, mc5=^P^R, mc4=^P^T,
```

640  AIX 5L Version 5.3 Files Reference
Files

/usr/share/lib/terminfo/?/*.  Compiled terminal capability database.

Related Information

The \texttt{captoinfo} command, \texttt{infocmp} command, \texttt{tic} command.

The \texttt{printf}, \texttt{fprintf}, or \texttt{sprintf} subroutine.

Curses Overview for Programming in \textit{AIX 5L Version 5.3 General Programming Concepts: Writing and Debugging Programs}.

\texttt{.tiprc} File Format for \texttt{tip}

\textbf{Purpose}

Provides initial settings of variables for the \texttt{tip} command.

\textbf{Description}

The \texttt{.tiprc} file allows you to initialize \texttt{variable settings} for the \texttt{tip} command. When first invoked the \texttt{tip} command searches the user’s home directory (defined by the \texttt{$HOME} environment variable) for a \texttt{.tiprc} file. If the file is present, the \texttt{tip} command sets the \texttt{tip} variables according to instructions in the \texttt{.tiprc} file.

The \texttt{tip} command uses several different types of variables: numeric, string, character, or Boolean. A Boolean variable can be toggled by putting the variable name in the \texttt{.tiprc} file, or it can be reset by putting an \texttt{!} (exclamation point) in front of the variable name. Other types of variables are set by following the variable name with an \texttt{=} (equal sign) and the new value of the variable.

You can use the \texttt{-v} flag of the \texttt{tip} command to see the variable settings as they are made. Also, you can use the \texttt{~s} escape signal to change variables while the \texttt{tip} command is running.

\textbf{Examples}

Following is a sample \texttt{.tiprc} file:

\begin{verbatim}
be
ba=9600
!echocheck
\end{verbatim}

This file toggles the \texttt{beautify} (be) variable, sets the \texttt{baudrate} (ba) variable to 9600, and resets the \texttt{echocheck} variable to the default setting.

\textbf{Files}

\texttt{$HOME/.tiprc}  Specifies the complete path name of the \texttt{.tiprc} file.

\textbf{Related Information}

The \texttt{tip} command.

The \texttt{Communication with connected UNIX systems using the tip command} in \textit{Networks and communication management}.
**trcfmt File Format**

**Purpose**
Stores trace templates.

**Description**
The `trcrpt` command, which formats trace reports, uses trace templates to determine how the data contained in trace entries should be formatted. All trace templates are stored in the master template file, `/etc/trcfmt`. Trace templates identify the trace hook ID, the version and release number, the indentation level, the event label, and data description fields. The data description fields contain formatting information for the trace entry data and can be repeated as many times as is necessary to format all of the trace data in the trace entry.

**Modifying this File**
The `trcfmt` file should only be modified using the `trcupdate` command. Trace hooks with values less than 010 are for internal use by the trace facilities. If these hooks are changed, the performance of trace, in particular `trcrpt`, is unpredictable.

**Trace Entries**
The data recorded for each traced event consist of a word containing the trace hook identifier and the hook type followed by a variable number of words of trace data optionally followed by a timestamp. The word containing the trace hook identifier and the hook type is call the hook word. The remaining two bytes of the hook word are called hook data and are available for recording event data.

*HookWord*  The first two bytes of a *HookWord* contain the *HookID* and *HookType*. The contents of the second two bytes depends on the value of the *HookType*.

*HookID*  The *HookID* is represented in the trace entry as 3 hexadecimal digits. For user programs, the hook id may be a value ranging from 0x010 to 0x0FF. *HookIDs* are defined in the `/usr/include/sys/trchkid.h` file.
**HookType**

The **HookType** is a 4-bit value that identifies the format of the remainder of the trace entry. You specify the **HookType** when you record the trace entry.

**Value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Trace Entry Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The trace entry consists of only the <strong>HookWord</strong>. The third and fourth bytes of the <strong>HookWord</strong> contain trace data. Trace entries of this type are recorded using the <strong>trchook</strong> or <strong>utrchook</strong> subroutine.</td>
</tr>
<tr>
<td>2</td>
<td>The trace entry consists of the <strong>HookWord</strong> and one additional word of trace data. The third and fourth bytes of the <strong>HookWord</strong> contain trace data. Trace entries of this type are recorded using the <strong>trchook</strong> or <strong>utrchook</strong> subroutine.</td>
</tr>
<tr>
<td>6</td>
<td>The trace entry consists of the <strong>HookWord</strong> and up to five additional words of trace data. The third and fourth bytes of the <strong>HookWord</strong> contain trace data. Trace entries of this type are recorded using the <strong>trchook</strong> or <strong>utrchook</strong> subroutine.</td>
</tr>
<tr>
<td>8</td>
<td>The trace entry consists of the <strong>HookWord</strong> and a data word followed by a variable number of bytes of trace data and a timestamp. The third and fourth bytes of the <strong>HookWord</strong> contain the number of bytes of trace data which follows the trace word. Trace entries of this type are recorded using the <strong>trcgent</strong> subroutine or the <strong>trcgenkt</strong> kernel service.</td>
</tr>
<tr>
<td>9</td>
<td>The trace entry consists of the <strong>HookWord</strong> and a timestamp. The third and fourth bytes of the <strong>HookWord</strong> contain trace data. Trace entries of this type are recorded using the <strong>trchook</strong> or <strong>utrchook</strong> subroutine.</td>
</tr>
<tr>
<td>A</td>
<td>The trace entry consists of the <strong>HookWord</strong>, one additional word of trace data, and a timestamp. The third and fourth bytes of the <strong>HookWord</strong> contain trace data. Trace entries of this type are recorded using the <strong>trchook</strong> or <strong>utrchook</strong> subroutine.</td>
</tr>
<tr>
<td>E</td>
<td>The trace entry consists of the <strong>HookWord</strong>, up to five additional words of trace data, and a timestamp. The third and fourth bytes of the <strong>HookWord</strong> contain trace data. Trace entries of this type are recorded using the <strong>trchook</strong> or <strong>utrchook</strong> subroutine.</td>
</tr>
<tr>
<td>0</td>
<td>The trace entry consists of the <strong>HookWord</strong> and a data word followed by a variable number of bytes of trace data. The third and fourth bytes of the <strong>HookWord</strong> contain the number of bytes of trace data which follows the trace word. Trace entries of this type are recorded using the <strong>trcgen</strong> subroutine or the <strong>trcgenk</strong> kernel service.</td>
</tr>
</tbody>
</table>

**Data Pointer**

The DATA POINTER is a pointer to the current position in the trace entry. The DATA POINTER is changed by the **trcrpt** as it interprets the template and formats the trace entry. The initial position of the DATA POINTER is the third byte of the **HookWord** for **HookTypes** 1, 9, 2, A, 6, and E and the first byte after the **HookWord** for **HookTypes** 0 and 8.

**Trace Data Formatting**

**Indentation Level**

The formatted trace data is aligned in columns corresponding to the source of the trace event. This is identified in each template using the **L=X** descriptor. The possible values of the **L=X** command are as follows:

- **L=APPL** Outputs the trace data in the APPL (application) column.
- **L=SVC** Outputs the trace data in the SVC (system call) column.
- **L=KERN** Outputs the trace data in the KERN (kernel) column.
- **L=INT** Outputs the trace data in the INT (interrupt) column.

**Continuation Character**

A \ (backslash) at the end of a line must be used to continue a template on the next line.
Labels or Text Strings
Individual strings (or labels) can be separated by any number of spaces or tabs, but all excess spacing is compressed to one blank on the trace report unless other format structures are put into effect. Labels are enclosed in double quotes (" ").

\n Outputs to a new line. Data on the new line is left-justified according to the value set in the INDENTATION LEVEL.
\t Inserts a tab. Tabs are expanded to spaces, using a fixed tabstop separation of 8.

Format Codes

DATA POINTER Position Format Codes
Gm.n Sets DATA POINTER to byte.bit location m.n.
Om.n Advances DATA POINTER by m.n byte.bits.
Rm Decrements DATA POINTER by m bytes.

Output Format Codes
Bm.n Sends output in Binary format where m is the length of the data in bytes and n is the length in bits. Unlike the other printing format codes, the DATA POINTER can be bit aligned and is not rounded up to the next byte boundary.
D2, D4, D8 Converts data to signed decimal format. The length of the data is two, four, or eight bytes, and the DATA POINTER is advanced by the same number of bytes.
F4 Converts data to C type ‘float’ floating point format. The length of the data is 4 bytes, and the DATA POINTER is advanced by 4 bytes.
F8 Converts data to C type ‘double’ floating point format. The length of the data is 8 bytes, and the DATA POINTER is advanced by 8 bytes.
S1, S2, S4 Left-justifies ASCII strings. The length of the string is in the first byte (half-word, word) of the data. The length of the string does not include this byte.
T4 Outputs the next 4 bytes as a date and time string.
U2, U4, U8 Converts data to unsigned decimal format. The length of the data is two, four, or eight bytes, and the DATA POINTER is advanced by the same number of bytes.
Xm Converts data to hexadecimal format. The DATA POINTER is advanced by m bytes.

Interpreter Format Codes
E1, E2, E4 Outputs the next byte (half_word, word) as an ‘errno’ value, replacing the numeric code with the corresponding #define name in the /usr/include/sys/errno.h file. The DATA POINTER is advanced by 1, 2, or 4 bytes.
P4 Uses the next word as a process ID, and outputs the pathname of the executable with that process ID. Process IDs and their pathnames are acquired by the trace command at the start of a trace and by the trcrpt command via a special EXEC tracehook. The DATA POINTER is advanced by 4 bytes.

Switch Statements
A SWITCH statement is a format code followed by a comma. Each CASE entry of the SWITCH statement consists of:
1. A ‘MatchValue’ with a type (usually numeric) corresponding to the format code.
2. A simple ‘String’ or a new ‘Descriptor’ bounded by braces. A descriptor is a sequence of format codes, strings, switches, and loops.
3. A comma delimiter.
The switch is terminated by a CASE entry without a comma delimiter. The CASE entry is selected as the first entry whose MatchValue is equal to the expansion of the format code. The special matchvalue ‘\*' is a wildcard and matches anything.

The DATA POINTER is advanced by the format code.

**LOOP Statements**

Loops are used to output binary buffers of data; therefore, the descriptor for a LOOP is usually X0 or X1. The syntax of a loop is LOOP format_code {descriptor}. The descriptor is executed \( N \) times, where \( N \) is the numeric value of the format code.

The DATA POINTER is advanced by the format code and by the operations of the descriptor.

**Macros**

Macros are temporary variables that work like shell variables. They are assigned a value with the syntax:

```bash
{{ $xxx = EXPRESSION }}
```

where EXPRESSION is a combination of format codes, macros, and constants. The operators + (addition), - (subtraction), / (division), and * (multiplication) are permissible within macros.

**Predefined Macros**

<table>
<thead>
<tr>
<th>Macro Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$BASEPOINTER</td>
<td>Marks the starting offset into an event. The default is 0, but the actual offset is the sum of the values of DATA POINTER and BASE_POINTER. It is used with template subroutines when the parts of an event have same structure and can be printed by same template but may have different starting points into an event.</td>
</tr>
<tr>
<td>$BREAK</td>
<td>Ends the current trace event.</td>
</tr>
<tr>
<td>$D1 - $D5</td>
<td>Dataword 1 through dataword 5. The DATA POINTER is not moved.</td>
</tr>
<tr>
<td>$DATAPOINTER</td>
<td>Activates the DATA POINTER. It can be set and manipulated like other user macros.</td>
</tr>
<tr>
<td>$DEFAULT</td>
<td>Uses the DEFAULT template 008.</td>
</tr>
<tr>
<td>$ERROR</td>
<td>Outputs an error message to the report and exit from the template after the current descriptor is processed. The error message supplies the logfile, the logfile offset of the start of that event, and the trace ID.</td>
</tr>
<tr>
<td>$EXECPATH</td>
<td>Outputs the pathname of the executable for the current process.</td>
</tr>
<tr>
<td>$HB</td>
<td>Number of bytes in trcgen subroutine variable length buffer. This is also equal to the 16-bit hook data.</td>
</tr>
<tr>
<td>$HD</td>
<td>Hook data (lower 16 bits).</td>
</tr>
<tr>
<td>$HT</td>
<td>Allows for multiple, different trchook subroutine call with the same template. The return values of the $HT macro are:</td>
</tr>
<tr>
<td></td>
<td><strong>Value</strong></td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>E</td>
</tr>
</tbody>
</table>

The DATA POINTER is not changed.

<p>| $L1-$L2         | Long (64-bit) dataword 1, or 2. For example, $L1 is the concatenation of $d1 and $d2. The 64-bit values would most likely have been traced with the TRCHK64L1 or TRCHK64L2 macros. No change to data pointer. |
| $LOGID0         | Current logfile offset at the start of the event.                         |</p>
<table>
<thead>
<tr>
<th>Macro Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$LOGIDX</td>
<td>Current logfile offset into this event.</td>
</tr>
<tr>
<td>$LOGFILE</td>
<td>Returns the name of the logfile being processed.</td>
</tr>
<tr>
<td>$MCR0, $MCR1, $MCR4</td>
<td>Machine MCR registers 0, 1, and A.</td>
</tr>
<tr>
<td>$PID</td>
<td>Outputs the current process ID.</td>
</tr>
<tr>
<td>$PMC1 - $PMC8</td>
<td>Machine PMC registers 1 through 8.</td>
</tr>
<tr>
<td>$PURR</td>
<td>Machine PURR register.</td>
</tr>
<tr>
<td>$RELLINENO</td>
<td>Line number for this event. The first line starts at 1.</td>
</tr>
<tr>
<td>$SKIP</td>
<td>Immediately ends a trace event without printing.</td>
</tr>
<tr>
<td>$STOP</td>
<td>Outputs the name of the current system call.</td>
</tr>
<tr>
<td>$PID</td>
<td>Outputs the current process ID.</td>
</tr>
<tr>
<td>$TRACEID</td>
<td>Returns the trace ID of the current event.</td>
</tr>
</tbody>
</table>

**Built-in Macros**

The built-in macros are:

- **buftofilename (bp)**: Looks up filename by buf struct.
- **fdinstall ( )**: Installs the file descriptor and the current v-node from lookuppn as a file_descriptor/v-node pair for this process ID.
- **fdtofilename ( )**: Looks up the filename for the given file descriptor for this process ID. If the filename is not found, nothing is output.
- **flih ( )**: Advances the Interrupt Depth.
- **lookuppnninstall1**: Installs the filename as the current file with the trcprt command.
- **lookuppnninstall2**: Install the v-node as the current v-node. It also installs the current_v-node/current_file as a v-node/filename par.
- **pfsrdwrinstall1 (vp)**: Sets the current v-node of this process to vp.
- **pfsrdwrinstall2 (VA,S, count)**: Creates a virtual address/v-node structure to be filled in be VMM hooks if a page fault occurs.
- **resume ( )**: Decrements the Interrupt Depth.
- **setdelim ( )**: Inhibits spaces between characters.
- **slihlookup ( )**: Looks up the second level interrupt handler.
- **sidtofilename (sid)**: Looks up filename by segment ID.
- **vmbufinstall ( )**: Looks up the v-node of the file through the virtual page/sid and install the v-node and buf as a v-node/bp pair. This will be used by lvm on down.
- **v-nodetofilename (vp)**: Looks up filenames by v-node.
- **vpagetofilename (vpage, sid)**: Looks up filenames by vpage and segment ID.

**Files**

- `/etc/trcfmt`: Stores trace templates.
- `/usr/include/sys/trchkid.h`: Defines hook identifiers.
- `/usr/include/sys/trcmacros.h`: Defines trace macros.

**Related Information**

The **trcupdate** command.

The **trcgen** subroutine, **trchook** subroutine.

Trace Facility Overview in *AIX 5L Version 5.3 General Programming Concepts: Writing and Debugging Programs.*
troff File Format

Purpose
Describes the output language from the troff command.

Description
The device-independent troff file format outputs a pure ASCII description of a typeset document. The description specifies the typesetting device, the fonts, and the point sizes of characters to be used, as well as the position of each character on the page.

A list of all the legal commands follows. Most numbers are denoted by the Number variable and are ASCII strings. Strings inside [ ] (brackets) are optional. The troff command can produce them, but they are not required for the specification of the language. The \n command character has the standard meaning of new-line character. Between commands, white space has no meaning. White-space characters are spaces and new lines.

The following are the legal commands:

- **s**Number
  Specifies the point size of the characters to be generated.

- **f**Number
  Indicates the font is to be mounted in the position specified by the Number variable value, which ranges from 0 (zero) to the highest font currently mounted. The 0 (zero) value is a special position, called by the troff command, but not directly accessible by the user. Fonts are normally mounted starting at position 1 (one).

- **c**Character
  Generates the specified character at the current location on the page; the value specified by the Character variable is a single-byte character.

- **C**XYZ
  Generates the XYZ special character whose name is delimited by white space. The name is one of the special characters legal for the typesetting device as specified in the DESC file. This file resides in a directory specific to the typesetting device. For instruction, see troff Font File Format and the /usr/lib/font/devDevice directory.

- **H**Number
  Changes the horizontal position on the page to the number specified. The number is in basic units of motions as specified by the DESC file. This is an absolute goto statement.

- **h**Number
  Adds the number specified to the current horizontal position. This is a relative goto statement.

- **V**Number
  Changes the vertical position on the page to the number specified (down is positive).

- **v**Number
  Adds the number specified to the current vertical position.

- **NumberCharacter**
  This is a two-digit number followed by an single-byte character. The meaning is a combination of the hNumber command followed by the cCharacter command. The specified number is added to the current horizontal position and then the single-byte character, specified by the Character variable, is produced. This is the most common form of character specification.

- **nB A**
  Indicates that the end of a line has been reached. No action is required, though by convention the horizontal position is set to 0 (zero). The troff command specifies a resetting of the x,y coordinates on the page before printing more characters. The first number, B, is the amount of space before the line and the second number, A, the amount of space after the line. The second number is delimited by white space.

- **w**
  A w command appears between words of the input document. No action is required. It is included so that one device can be emulated more easily on another device.

- **p**Number
  Begins a new page. The new page number is included in this command. The vertical position on the page should be set to 0 (zero).

- **#...\n**
  Initiates a comment line with the # (pound sign).
\textbf{Dl} \texttt{X Y}

Draws a line from the current position to that specified by the \texttt{X,Y} variables.

\textbf{Dc} \texttt{Dn}

Draws a circle of the diameter specified by the \texttt{D} variable with the leftmost edge being at the current location \((X,Y)\). The current location after drawing the circle is \(X+D,Y\), the rightmost edge of the circle.

\textbf{De} \texttt{DX DY n}

Draws an ellipse with the specified axes. The \texttt{DX} variable is the axis in the \texttt{X} direction and the \texttt{DY} variable is the axis in the \texttt{Y} direction. The leftmost edge of the ellipse is at the current location. After drawing the ellipse, the current location is \(X+DX,Y\).

\textbf{Da} \texttt{DH1 DV1 DH2 DV2 n}

Draws a counterclockwise arc from the current position to the \(\texttt{DH1}+\texttt{DH2},\ \texttt{DV1}+\texttt{DV2}\) variable that has a center of \(\texttt{DH1},\ \texttt{DV1}\) from the current position. The current location after drawing the arc is at its end.

\textbf{D~} \texttt{X Y X Y ... n}

Draws a spline curve (wiggly line) between each of the \texttt{X,Y} coordinate pairs starting at the current location. The final location is the final \texttt{X,Y} pair of the list.

\textbf{x P[aper]} \texttt{PaperSize W L n}

Specifies the name of the paper size to be printed. Valid paper sizes are Letter, Legal, A4, B5, Executive, and A5, where \texttt{W} and \texttt{L} are the paper width and length in machine units.

\textbf{x i[nit]n}

Initializes the typesetting device. The actions required are dependent on the device. An initializing command always occurs before any output generation is attempted.

\textbf{x T} \texttt{Device n}

Specifies the name of the typesetter with the \texttt{Device} variable. This is the same as the variable to the \texttt{-T} flag. Information about the typesetter is found in the \texttt{/usr/lib/font/devDevice} directory.

\textbf{x r[es]} \texttt{N H V n}

Specifies the resolution of the typesetting device in increments per inch with the \texttt{N} variable. The \texttt{H} variable specifies units of basic increments that horizontal motion will take place. The \texttt{V} variable indicates the units of basic increments for vertical motion.

\textbf{x p[ause]n}

Pauses the process by causing the current page to finish but does not relinquish the typesetter.

\textbf{x s[top]n}

 Stops the process by causing the current page to finish and then relinquishes the typesetter. Performs any shutdown and bookkeeping procedures required.

\textbf{x t[railer]n}

 Generates a trailer. On some devices, no operation is performed.

\textbf{x f[ont]} \texttt{N Font n}

 Loads the specified font into position \texttt{N}.

\textbf{x H[eight]} \texttt{M n}

 Sets the character height to \texttt{N} points. This causes the letters to be elongated or shortened. It does not affect the width of a letter. Not all typesetters can do this.

\textbf{x S[lant]} \texttt{M n}

 Sets the slant to \texttt{N} degrees. Only some typesetters can do this and not all angles are supported.

\textbf{x c[odeset]} \texttt{CS n}

 Switch to codeset \texttt{CS}. For example:

\begin{verbatim}
  x codeset ISO8859-1
\end{verbatim}

The following commands are effective on multi-byte characters.

\textbf{Q} \texttt{C1C2}

Outputs the character specified by the 2 bytes specified by the \texttt{C1} and \texttt{C2} variables. The high-order bits can be set in these bytes.

\textbf{R} \texttt{C1C2C3}

Outputs the character specified by the three bytes of the \texttt{C1}, \texttt{C2}, and \texttt{C3} parameters. The high-order bits can be set in these bytes.

\textbf{S} \texttt{C1C2C3C4}

Outputs the character specified by the four bytes of the \texttt{C1}, \texttt{C2}, \texttt{C3}, and \texttt{C4} parameters. The high-order bits can be set in these bytes.

\section*{Files}

\texttt{/usr/lib/font/devDevice}

Contains the \texttt{DESC} file and phototypesetter-specific files.

648  
AIX 5L Version 5.3 Files Reference
Related Information

International character support in text formatting in *Operating system and device management* discusses the European-language extended character set and the commands that use it.

The **troff Font** File Format.

The **troff** command.

---

**troff Font File Format**

**Purpose**

Specifies description files for the **troff** command.

**Description**

For each phototypesetter that the **troff** command supports and that is available on your system, there is a directory that contains files describing the phototypesetter and its fonts. This directory is named `/usr/lib/font/dev/Name`, where the *Name* variable specifies the name of the phototypesetter.

The ASCII **DESC** file in the `/usr/lib/font/dev/Name` directory within the **troff** command source directory describes the characteristics of the phototypesetter specified by the *Name* variable. A binary version of this file is found in the `/usr/lib/font/dev/Name/DESC.out` file. Each line of this ASCII file starts with a word that identifies a characteristic, followed by appropriate specifiers. Blank lines and lines beginning with the `#` (pound sign) are ignored.

For many typesetters, downloaded fonts are supported in a general fashion. The bitmaps for these fonts are stored in the `/usr/lib/font/dev/Name/bitmaps` directory. Each font size pair is stored in a file with a name of the form *Fontname-Size.pk*. For example:

```
B-24.pk
```

These bitmaps are stored in the PK packed-font format used by TeX and its post-processors. These bitmaps are easily generated from readily available programs, such as METAFONT, or easily converted from other forms.

In addition to the bitmap files, a **troff** font file, as described here, is required for each font typeface. In the `unitwidth` field of this file, the width of each character bitmap in device units is given.

The legal lines for the **DESC** file are:

- **res Number**
  - Resolution of device in basic increments per inch.
- **unitwidth Number**
  - Point size in which all width tables in the font description files are given. The **troff** command automatically scales the widths from the **unitwidth** size to the point size with which it is working.
- **sizescale Number**
  - Scaling for fractional point sizes. The value of the *Number* variable is 1. The **sizescale** line is not currently used.
- **paperwidth Number**
  - Width of paper in basic increments.
- **paperlength Number**
  - Length of paper in basic increments.
- **biggestfont Number**
  - Maximum number of characters in a font.
- **sizes Number1 Number2...**
  - List of point sizes available on typesetter, ended by 0.
- **fonts NumberName...**
  - Number of initial fonts, followed by the ASCII names of the fonts. For example:
    ```
    fonts 4 R I B S
    ```

---
**codeset codesetName**

Code set for the particular printer or typesetter, where *CodesetName* is a valid code set name for use with the `iconv` command. The specified code set is used to define character entries in the charset section of font description files. For example:

```
codeset ISO8859-1
```

The `troff` command uses the specified *CodesetName* and the code set implied by the current locale to determine if code set conversions are necessary for the input characters. The `iconv` function is used to perform the code set conversion if necessary.

**charset**

Last keyword in the file is on a line by itself. Following it is the list of special character names for this device. Names are separated by a space or a new line. The list can be as long as necessary. Names not in this list are not allowed in the font description files.

**hor Number**

Smallest unit of horizontal motion.

**vert Number**

Smallest unit of vertical motion.

The `hor` and `vert` lines describe the relationships between motions in the horizontal and vertical directions. For example, if the device moves in single basic increments in both directions, both the `hor` and `vert` lines have values of 1. If vertical motion occurs only in multiples of two basic units and horizontal motion occurs only in one basic unit, `vert` is 2 and `hor` is 1.

For each font supported by the phototypesetter, there is also an ASCII file with the same name as the font (for instance, `R`, `I`, `CW`) that describes it. The format for a font description file is as follows:

```
name Name
internalname Name
special
ligatures Name...0
spacewidth Number
charset
```

Describes one character in the font. Each line has one of two formats:

```
Name Width Kerning Code
```

OR

```
Name "
```

where the value of the `Name` field is either a single-byte character or a special character name from the list found in the `DESC` file. The `Width` field is in basic increments. The `Kerning` field is 1 if the character descends below the line, 2 if it rises above the letter ‘a’, and 3 if it both rises and descends. The `Code` field is the number sent to the typesetter to produce the character. For an `nls` font, the `Code` field can be a multi-byte sequence.

For fonts of extended-character output devices, the `Code` field can be a multi-byte sequence that begins and ends with a double quotation mark. In the sequence, control or nonprinting characters can be represented by the following escape sequences:

```
\n Produces a new line.
\r Produces a return.
\t Produces a tab.
\b Produces a backspace.
\" Produces a double quote.
\xd Produces a hexadecimal number, where dd is two hexadecimal digits.
```
Produces an octal number, where \texttt{ooo} is three octal digits.

The second format, \texttt{Name ""}, is used to show that the character has more than one name. The double quotation marks indicate that this name has the same values as the preceding line. The \texttt{Kerning} and \texttt{Code} fields are not used if the value of the \texttt{Width} field is a double quotation mark. The total number of different characters in this list should not be greater than the value of the \texttt{biggestfont} line in the DESC file.

The \texttt{DESC.out} and \texttt{Font.out} files were created as a result of executing the \texttt{makedev} program on the DESC file.

Prototype characters are provided for the charset section of the font table for fonts in large-character sets. Most characters in large-character sets, such as the Japanese, Chinese, and Korean character sets, have the same width. These prototype characters specify the width of characters with varying byte lengths. The \texttt{Kerning} and \texttt{Code} fields are not available for prototype character entries. These entries apply to all characters not explicitly defined in the charset section. It is assumed that the printer or typesetter code for characters handled through prototype characters is the same as the input code for the character after conversion by the \texttt{iconv} function. The following are the prototype character definitions:

<table>
<thead>
<tr>
<th>X0</th>
<th>Width</th>
<th>Width of all characters that return a value of 0 for \texttt{csid()}.</th>
</tr>
</thead>
<tbody>
<tr>
<td>X1</td>
<td>Width</td>
<td>Width of all 1-byte characters not defined elsewhere.</td>
</tr>
<tr>
<td>X2</td>
<td>Width</td>
<td>Width of all 2-byte characters not defined elsewhere.</td>
</tr>
<tr>
<td>(X_i)</td>
<td>Width</td>
<td>Width of all characters that return a value of (i) for \texttt{csid()}.</td>
</tr>
<tr>
<td>X3</td>
<td>Width</td>
<td>Width of all 3-byte characters not defined elsewhere.</td>
</tr>
<tr>
<td>X4</td>
<td>Width</td>
<td>Width of all 4-byte characters not defined elsewhere.</td>
</tr>
</tbody>
</table>

For example, the following prototype character definitions apply to the Japanese character sets (both IBM-932 and IBM-eucJP):

- \(X0\) : alphanumerics
- \(X1\) : JIS level 1 and 2 Kanji characters in JISX0208.1990
- \(X2\) : Katakana characters
- \(X3\) : IBM selected characters

Files

/\texttt{usr/lib/font/dev\texttt{Name}/DESC.out} file
Contains the description file for phototypesetter specified by the \texttt{Name} variable.

/\texttt{usr/lib/font/dev\texttt{Name}/bitmaps} directory
Contains bitmap files.

/\texttt{usr/lib/font/dev\texttt{Name}/Font.out} file
Contains the font description file for phototypesetter specified by the \texttt{Name} variable.

Related Information

The \texttt{troff} file format.

The \texttt{troff} command.

The \texttt{iconv} subroutine.
tunables File Format

Purpose
Centralizes tunable parameter values.

Description
Tunables files contain one or more sections, called “stanzas”. A stanza is started by a line containing the stanza name followed by a colon (:). There is no marking for the end of a stanza. It simply continues until another stanza starts. Each stanza contains a set of parameter/value pairs; one pair per line. The values are surrounded by double quotes ("), and an equal sign (=) separates the parameter name from its value. A parameter/value pair must necessarily belong to a stanza. It has no meaning outside of a stanza. Two parameters sharing the same name but belonging to different stanzas are considered to be different parameters. If a parameter appears several times in a stanza, only its first occurrence is used. Following occurrences are simply ignored. Similarly, if a stanza appears multiple times in the file, only the first occurrence is used. Everything following a number sign (#) is considered a comment and ignored. Heading and trailing blanks are also ignored.

A tunable file uses the following syntax:

```plaintext
# first stanza
stanza1:
  param1 = "value1"
  param2 = "value2"
  param2 = "value3"  # ignored, since already defined

# another stanza
stanza2:
  param1 = "value4"  # not the same parameter as param1 in stanza1

# the first stanza again
stanza1:  # ignored since already defined
```

Tunables files currently support seven different stanzas: one for each of the tunable commands (schedo, vmo, ioo, raso, no and nfso), plus a special info stanza. The six stanzas, schedo, vmo, ioo, raso, no and nfso, contain tunable parameters managed by the corresponding command (see the command man pages for the complete parameter lists).

The value can either be a numerical value or the literal words DEFAULT, which is interpreted as this tunable’s default value, and STATIC, which indicates a Static variable which is never restored. It is possible that some stanza contains values for non-existent parameters (in the case a tunable file was copied from a machine running an older version of AIX and one or more tunables do not exist anymore). Both the tunrestore and the tuncheck commands will print warnings about such parameters.

The info stanza is used to store information about the purpose of the tunable file and the level of AIX on which it was validated. Any parameter is acceptable in this stanza, however, some fields have a special meaning:

- **Description**: A character string describing the tunable file. SMIT displays this field in the file selection box.
- **AIX_level**: AIX version. This field is automatically updated by tunsave and tuncheck (on success only).
- **Kernel_type**: "UP" this is a uniprocessor kernel.
  "MP" this is a multiprocessor kernel.
  "MP64" this is a 64 bits multiprocessor kernel.
  This field is automatically updated by tunsave and tuncheck (on success only).
Last_validation The date this file was validated for the last time, and the type of validation:
'current' the file has been validated against the current context.
'reboot' the file has been validated against the nextboot context.
This field is automatically updated by **tunsave** and **tuncheck** (on success only).

Logfile_checksum The checksum of the lastboot.log file matching this tunables file. This field is present only in the lastboot file.

Other stanzas like info, schedo, vmo, ioo, raso, no and nfso may be present. These stanzas are simply ignored by the tunrestore command, but flagged by the tuncheck command.

Three files under /etc/tunables have special names and meaning:

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nextboot</td>
<td>This file is automatically applied at boot time. The bosboot command also get the value of Bosboot types tunables from this file. It contains all tunable settings made permanent.</td>
</tr>
<tr>
<td>lastboot</td>
<td>This file is automatically generated at boot time. It contains the full set of tunable parameters, with their values after the last boot. Default values are marked with # DEFAULT VALUE.</td>
</tr>
<tr>
<td>lastboot.log</td>
<td>This should be the only file in /etc/tunables that is not in the stanza format described here. It is automatically generated at boot time, and contains the logging of the creation of the lastboot file, i.e. any parameter change made is logged. Any change which could not be made (possible if the nextboot file was created manually and not validated with tuncheck) is also logged.</td>
</tr>
</tbody>
</table>

Examples
The following is a sample tunables file:

```
info:
    Description = "Set of tunables for departmental server"
    AIX_level = "5.2.0.0"
    Kernel_type = "UP"
    Last_validation = "2002-06-16 12:11:11 CDT current"

schedo:
    timeslice = "2" # set timeslice to 30ms
    sched_D = "DEFAULT" # value was 123

vmo:
    minperm = "48538"
    memory_frames = "65536" # STATIC (never restored)

ioo:
    iotunable = "value"

no:
    ipforwarding = "1"
    ipsrcrouteforward = "1"
    thewall = "STATIC" # value was 131072 (never restored)

nfso:
    nfs_allow_all_signals = "0" # DEFAULT VALUE
    nfs_device_specific_bufs = "0"
```

Files
All the tunable files are located in the /etc/tunables directory.

- /etc/tunables/nextboot Contains the values to be applied at the next rebooting of the machine.
- /etc/tunables/lastboot Contains the values for all tuning parameters after the last rebooting of the machine.
Related Information
The `schedo`, `vmo`, `ioo`, `raso`, `no`, `tunchange`, `tndefault`, `tunsave`, `tunrestore`, `tuncheck` and `nfso` commands.

Kernel Tuning in the AIX 5L Version 5.3 Performance Tools Guide and Reference.

uconvdef Source File Format

Purpose
Defines UCS-2 (Unicode) conversion mappings for input to the `uconvdef` command.

Description
Conversion mapping values are defined using UCS-2 symbolic character names followed by character encoding (code point) values for the multibyte code set. For example,

```
<U0020> \x20
```

represents the mapping between the `<U0020>` UCS-2 symbolic character name for the space character and the `\x20` hexadecimal code point for the space character in ASCII.

In addition to the code set mappings, directives are interpreted by the `uconvdef` command to produce the compiled table. These directives must precede the code set mapping section. They consist of the following keywords surrounded by `< >` (angle brackets), starting in column 1, followed by white space and the value to be assigned to the symbol:

- `<code_set_name>` The name of the coded character set, enclosed in quotation marks (" "), for which the character set description file is defined.
- `<mb_cur_max>` The maximum number of bytes in a multibyte character. The default value is 1.
- `<mb_cur_min>` An unsigned positive integer value that defines the minimum number of bytes in a character for the encoded character set. The value is less than or equal to `<mb_cur_max>`. If not specified, the minimum number is equal to `<mb_cur_max>`.
- `<escape_char>` The escape character used to indicate that the character following is interpreted in a special way. This defaults to a backslash (\).
- `<comment_char>` The character that, when placed in column 1 of a `charmap` line, is used to indicate that the line is ignored. The default character is the number sign (#).
- `<char_name_mask>` A quoted string consisting of format specifiers for the UCS-2 symbolic names. This must be a value of AXXXX, indicating an alphabetic character followed by 4 hexadecimal digits. Also, the alphabetic character must be a U, and the hexadecimal digits must represent the UCS-2 code point for the character. An example of a symbolic character name based on this mask is `<U0020>` Unicode space character.
<uconv_class> Specifies the type of the code set. It must be one of the following:
  SBCS  Single-byte encoding
  DBCS  Stateless double-byte, single-byte, or mixed encodings
  EBCDIC_STATEFUL  Stateful double-byte, single-byte, or mixed encodings
  MBCS  Stateless multibyte encoding

This type is used to direct uconvdef on what type of table to build. It is also stored in the table to indicate the type of processing algorithm in the UCS conversion methods.

<locale> Specifies the default locale name to be used if locale information is needed.

<subchar> Specifies the encoding of the default substitute character in the multibyte code set.

The mapping definition section consists of a sequence of mapping definition lines preceded by a CHARMAP declaration and terminated by an END CHARMAP declaration. Empty lines and lines containing <comment char> in the first column are ignored.

Symbolic character names in mapping lines must follow the pattern specified in the <char_name_mask>, except for the reserved symbolic name, <unassigned>, that indicates the associated code points are unassigned.

Each noncomment line of the character set mapping definition must be in one of the following formats:

1. "%s %s %s/n", <symbolic-name>, <encoding>, <comments>
   For example:
   <U3004>   \x81\x57

   This format defines a single symbolic character name and a corresponding encoding.
   The encoding part is expressed as one or more concatenated decimal, hexadecimal, or octal constants in the following formats:
   • "%cd%d", <escape_char>, <decimal byte value>
   • "%cx%x", <escape_char>, <hexadecimal byte value>
   • "%c%o", <escape_char>, <octal byte value>

   Decimal constants are represented by two or more decimal digits preceded by the escape character and the lowercase letter d, as in \d97 or \d143. Hexadecimal constants are represented by two or more hexadecimal digits preceded by an escape character and the lowercase letter x, as in \x61 or \x8f. Octal constants are represented by two or more octal digits preceded by an escape character.
   Each constant represents a single-byte value. When constants are concatenated for multibyte character values, the last value specifies the least significant octet and preceding constants specify successively more significant octets.

2. "%s ...%s %s/n", <symbolic-name>, <symbolic-name>, <encoding>, <comments>
   For example:
   <U3003>...<U3006>   \x81\x56

   This format defines a range of symbolic character names and corresponding encodings. The range is interpreted as a series of symbolic names formed from the alphabetic prefix and all the values in the range defined by the numeric suffixes.
   The listed encoding value is assigned to the first symbolic name, and subsequent symbolic names in the range are assigned corresponding incremental values. For example, the line:
   <U3003>...<U3006>   \x81\x56
is interpreted as:

\textcode{\textless U3003> \textbackslash x81\textbackslash x56}
\textcode{\textless U3004> \textbackslash x81\textbackslash x57}
\textcode{\textless U3005> \textbackslash x81\textbackslash x58}
\textcode{\textless U3006> \textbackslash x81\textbackslash x59}

3. "<unassigned> %s . .%s %s/n", <encoding>, <encoding>, <comments>
   
   This format defines a range of one or more unassigned encodings. For example, the line:
   
   \textcode{<unassigned> \textbackslash x9b...\textbackslash x9c}

   is interpreted as:
   
   \textcode{<unassigned> \textbackslash x9b}
   \textcode{<unassigned> \textbackslash x9c}

**Related Information**

The \texttt{uconvdef} command.

[Code Set Overview](AIX 5L Version 5.3 Kernel Extensions and Device Support Programming Concepts)

[List of UCS-2 Interchange Converters](AIX 5L Version 5.3 General Programming Concepts: Writing and Debugging Programs)

---

**UIL File Format**

**Purpose**

Contains information on the user interface for a widget-based application.

**Description**

User Interface Language (UIL) is used to describe the initial state of a user interface for a widget-based application. UIL describes the widgets used in the interface, the resources of those widgets, and the callbacks of those widgets. A UIL file is compiled into a user interface definition (UID) file using the \texttt{uil} command or the \texttt{Uil} callable compiler function. The contents of the compiled UID file can then be accessed by the various Motif Resource Manager (MRM) functions from within an application program.

The syntax for the UIL is as follows:

\texttt{MODULE [ModuleName]}
\texttt{[NAMES = CASE_INSENSITIVE | CASE_SENSITIVE ]}
\texttt{[CHARACTER_SET = CharacterSet ]}
\texttt{[OBJECTS = { WidgetName = GADGET | WIDGET; [... ] } ]}
\texttt{[ ... ]}
\texttt{END MODULE;}

---

656 AIX 5L Version 5.3 Files Reference
File Format
UIL is a free-form language. This means that high-level constructs, such as object and value declarations, do not need to begin in any particular column and can span any number of lines. Low-level constructs, such as keywords and punctuation characters, can also begin in any column; however, except for string literals and comments, they cannot span lines.

The UIL compiler accepts input lines up to 132 characters in length.

**MODULE ModuleName**
The name by which the UIL module is known in the UID file. This name is stored in the UID file for later use in the retrieval of resources by the MRM. This module name is always uppercase.

**NAMES = CASE_INSENSITIVE | CASE_SENSITIVE**
Indicates whether names should be treated as case-sensitive or case-insensitive. The default is case-sensitive. The case-sensitivity clause should be the first clause in the module header and must precede any statement that contains a name. If names are case-sensitive in a UIL module, UIL keywords in that module must be in lowercase. Each name is stored in the UID file in the same case as it appears in the UIL module. If names are case-insensitive, keywords can be in uppercase, lowercase, or mixed case, and the uppercase equivalent of each name is stored in the UID file.

**CHARACTER_SET = CharacterSet**
Specifies the default character set for string literals in the module that do not explicitly set their character set. In the absence of this clause, the default character set is the codeset component of the `LANG` environment variable, or the value of `XmFALLBACK_CHARSET` if `LANG` is not set or has no codeset component. The value of `XmFALLBACK_CHARSET` is defined by the UIL supplier, but is usually `ISO8859-1` (equivalent to `ISO_LATIN1`). Use of this clause turns off all localized string literal processing turned on by either the `-s` compiler flag or the `Uil_command_type` data structure element `use_setlocale_flag`.

**OBJECTS = { WidgetName = GADGET | WIDGET;]**
Indicates whether the widget or gadget form of the control specified by `WidgetName` variable is used by default. The widget form is used by default. The specified control should be one that has both a widget and gadget version, for example: `XmCascadeButton`, `XmLabel`, `XmPushButton`, `XmSeparator`, and `XmToggleButton`. The form of more than one control can be specified by delimiting them with `;` (semicolons). The gadget or widget form of an instance of a control can be specified with the `GADGET` and `WIDGET` keywords in a particular object declaration.

**ValueSection**
Provides a way to name a value expression or literal. The value name can then be referred to by declarations that occur elsewhere in the UIL module in any context where a value can be used. Values can be forward-referenced. See [Value Sections](#) for more detail.

**ProcedureSection**
Defines the callback functions used by a widget and the creation functions for user-defined widgets. These definitions are used for error checking. See [Procedure Sections](#) for more detail.
ListSection
Provides a way to group together a set of arguments, controls (children), callbacks, or procedures for later use in the UIL module. Lists can contain other lists so you can set up a hierarchy to clearly show which arguments, controls, callbacks, and procedures are common to which widgets. See "List Sections" for more detail.

ObjectSection
Defines the objects that make up the user interface of the application. You can reference the object names in declarations that occur elsewhere in the UIL module in any context where an object name can be used (for example, in a controls list, as a symbolic reference to a widget ID, or as the TagValue argument for a callback procedure). Objects can be forward-referenced. See "Object Sections" for more detail.

IdentifierSection
Defines a run-time binding of data to names that appear in the UIL module. See "Identifier Sections" for more detail.

The UIL file can also contain comments and include directives. These, as well as the main elements of the UIL file format, are described in the following sections.

Comments
Comments can take one of two forms, neither of which can be nested:
- The comment is introduced with the /* sequence followed by the text of the comment and terminated with the */ sequence. This form of comment can span multiple source lines.
- The comment is introduced with an ! (exclamation point) followed by the text of the comment and terminated by the end of the source line.

Value Sections
A value section consists of the VALUE keyword followed by a sequence of value declarations. It has the following syntax:

```
VALUE ValueName :
[ EXPORTED | PRIVATE ] ValueExpression |
IMPORTED ValueType ;
```

ValueExpression is assigned to ValueName, or a ValueType is assigned to an imported value name. A value declaration provides a way to name a value expression or literal. The value name can be referred to by declarations that occur later in the UIL module in any context where a value can be used. Values can be forward-referenced.

EXPORTED A value that you define as exported is stored in the UID file as a named resource and can be referenced by name in other UID files. When you define a value as exported, MRM looks outside the module in which the exported value is declared to get its value at run time.

PRIVATE A private value is a value that is not imported or exported. A value that you define as private is not stored as a distinct resource in the UID file. You can reference a private value only in the UIL module containing the value declaration. The value or object is directly incorporated into anything in the UIL module that references the declaration.

IMPORTED A value that you define as imported is one that is defined as a named resource in a UID file. MRM resolves this declaration with the corresponding exported declaration at application run time.

By default, values and objects are private. The following is a list of the supported value types in UIL:

- ANY
- ARGUMENT
- BOOLEAN
- COLOR
Procedure Sections
A procedure section consists of the PROCEUDRE keyword followed by a sequence of procedure declarations. It has the following syntax:

**PROCEDURE**

*ProcedureName* [ ([ ValueType ] ) ] ;

Use a procedure declaration to declare the following:

- A function that can be used as a callback function for a widget
- The creation function for a user-defined widget.

You can reference a procedure name in declarations that occur later in the UIL module in any context where a procedure can be used. Procedures can be forward-referenced. You cannot use a name that you used in another context as a procedure name.

In a procedure declaration, you have the option of specifying that a parameter is passed to the corresponding callback function at run time. This parameter is called the *callback tag*. You can specify the data type of the callback tag by putting the data type in parentheses following the procedure name. When you compile the module, the UIL compiler checks that the argument you specify in references to the procedure is of this type. Note that the data type of the callback tag must be one of the valid UIL data types. You can use a widget as a callback tag, as long as the widget is defined in the same widget hierarchy as the callback; that is, they must have a common ancestor that is in the same UIL hierarchy.

The following list summarizes how the UIL compiler checks argument type and argument count, depending on the procedure declaration:

- **No parameters**
  - No argument type or argument count checking occurs. You can supply either 0 or 1 arguments in the procedure reference.
  - ()
    - Checks that the argument count is 0.
  - (ANY)
    - Checks that the argument count is 1. Does not check the argument type. Use the ANY data type to prevent type checking on procedure tags.
  - (Type)
    - Checks for one argument of the specified type.
  - (ClassName)
    - Checks for one widget argument of the specified widget class.
While it is possible to use any UIL data type to specify the type of a tag in a procedure declaration, you must be able to represent that data type in the programming language you are using. Some data types (such as integer, Boolean, and string) are common data types recognized by most programming languages. Other UIL data types (such as string tables) are more complicated and may require you to set up an appropriate corresponding data structure in the application in order to pass a tag of that type to a callback function.

You can also use a procedure declaration to specify the creation function for a user-defined widget. In this case, you specify no formal parameters. The procedure is called with the standard three arguments passed to all widget creation functions. See "Chapter 1. AIXwindows Overview for Programmers" in AIX 5L Version 5.3 AIXwindows Programming Guide for more information about widget creation functions.

List Sections
A list section consists of the LIST keyword followed by a sequence of list declarations. It has the following syntax:

```
LIST
ListName : { ListItem; [...] }
[...
```

You can also use list sections to group together a set of arguments, controls (children), callbacks, or procedures for later use in the UIL module. Lists can contain other lists so you can set up a hierarchy to clearly show which arguments, controls, callbacks, and procedures are common to which widgets. You cannot mix the different types of lists; a list of a particular type cannot contain entries of a different list type or reference the name of a different list type. A list name is always private to the UIL module in which you declare the list and cannot be stored as a named resource in a UID file.

The additional list types are described in the following sections.

**Arguments List Structure:** An arguments list defines which arguments are specified in the arguments-list parameter when the creation function for a particular object is called at run time. An arguments list also specifies the values for those arguments. Arguments lists have the following syntax:

```
LIST ListName : ARGUMENTS {
ArgumentName = ValueExpression;
[...] }
[...
```

The argument name (ArgumentName) must be either a built-in argument name or a user-defined argument name that is specified with the ARGUMENTS function.

If you use a built-in argument name as an arguments list entry in an object definition, the UIL compiler checks the argument name to be sure that it is supported by the type of object that you are defining. If the same argument name is displayed more than once in a given arguments list, the last entry that uses that argument name supersedes all previous entries with that name, and the compiler issues a message.

Some arguments, such as XmNitems and XmNitemCount, are coupled by the UIL compiler. When you specify one of the coupled arguments, the compiler also sets the other one. The coupled argument is not available to you.

AIXwindows and the X Toolkit (Intrinsics) support constraint arguments. A constraint argument is one that is passed to children of an object, beyond those arguments normally available. For example, the Form widget grants a set of constraint arguments to its children. These arguments control the position of the children within the Form widget.
Unlike the arguments used to define the attributes of a particular widget, constraint arguments are used exclusively to define additional attributes of the children of a particular widget. These attributes affect the behavior of the children within their parent. To supply constraint arguments to the children, include the arguments in the arguments list for the child.

**Callbacks List Structure:** Use a callbacks list to define which callback reasons are to be processed by a particular widget at run time. Callback lists have the following syntax:

```
LIST
ListName : CALLBACKS {
    ReasonName = PROCEDURE ProcedureName [ [ ValueExpression ] ]; |
    ReasonName = ProcedureList ;
[...]
[...]
```

For AIXwindows widgets, the reason name must be a built-in reason name. For a user-defined widget, you can use a reason name that you previously specified using the REASON function. If you use a built-in reason in an object definition, the UIL compiler ensures that reason is supported by the type of object you are defining.

If the same reason is displayed more than once in a callbacks list, the last entry referring to that name supersedes all previous entries using the same reason. The UIL compiler then issues a diagnostic message.

If you specify a named value for the procedure argument (callback tag), the data type of the value must match the type specified for the callback tag in the corresponding procedure declaration. When specifying a widget name as a procedure value expression, you must also specify the type of the widget and a space before the name of the widget.

Because the UIL compiler produces a UID file rather than an object module (.o), the binding of the UIL name to the address of the entry point and then to the procedure is not done by the loader. Instead, this binding is established at run time with the MRM function. You call this function before fetching any objects, giving it both the UIL names and the procedure addresses of each callback. The name you register with MRM in the application program must match the name you specified for the procedure in the UIL module.

Each callback procedure received three arguments. The first two arguments have the same form for each callback. The form of the third argument varies from object to object.

The first argument is the address of the data structure maintained by the AIXwindows for this object instance. This address is called the widget ID for this object.

The second argument is the address of the value you specified in the callbacks list for this procedure. If you do not specify an argument, the address is null.

The third argument is the reason name you specified in the callbacks list.

**Controls List Structure:** A controls list defines which objects are children of, or controlled by, a particular object. Each entry in a controls list has the following syntax:

```
LIST
ListName : CONTROLS {
    [ ChildName ] [ MANAGED | UNMANAGED ] ObjectDefinition;
[...]
[...]
```
If you specify the `MANAGED` keyword at run time, the object is created and managed; if you specify the `UNMANAGED` keyword at run time, the object is only created. Objects are managed by default.

You can use the `ChildName` parameter to specify resources for the automatically created children of a particular control. Names for automatically created children are formed by appending `Xm_` to the name of the child widget. This name is specified in the documentation for the parent widget.

Unlike the arguments list and the callbacks list, a controls list entry that is identical to a previous entry does not supersede the previous entry. At run time, each controls list entry causes a child to be created when the parent is created. If the same object definition is used for multiple children, multiple instances of the child are created at run time.

**Procedures List Structure:** You can specify multiple procedures for a callback reason in UIL by defining a procedures list. Just as with other list types, procedures lists can be defined in-line or in a list section and referenced by name.

If you define a reason more than once (for example, when the reason is defined both in a referenced procedures list and in the callbacks list for the object), previous definitions are overridden by the latest definition. The syntax for a procedures list is as follows:

```
LIST
ListName : PROCEDURES {
ProcedureName [ ( [ ValueExpression ] ) ];
[...] }
[...]
```

When specifying a widget name as a procedure value expression, you must also specify the type of the widget and a space before the name of the widget.

**Object Sections**

An object section consists of the `OBJECT` keyword followed by a sequence of object declarations. It has the following syntax:

```
OBJECT ObjectName :
[ EXPORTED | PRIVATE | IMPORTED ] ObjectType
[ PROCEDURE CreationFunction ]
[ ObjectName [ WIDGET | GADGET ] ] { ListDefinitions }
```

Use an object declaration to define the objects that are stored in the UID file. You can reference the object name in declarations that occur elsewhere in the UIL module in any context where an object name can be used (for example, in a controls list, as a symbolic reference to a widget ID, or as the `TagValue` argument for a callback procedure). Objects can be forward-referenced, meaning that you can declare an object name after you have referenced it. All references to an object name must be consistent with the type of the object, as specified in the object declaration. You can specify an object as exported, imported, or private.

The object definition can contain a sequence of lists that define the arguments, hierarchy, and callbacks for the widget. You can only specify one list of each type for an object. When you declare a user-defined widget, you must include a reference to the widget creation function for the user-defined widget.

Use the `GADGET` or `WIDGET` keyword to specify the object type or to override the default variant for this object type. You can use the AIXwindows name of an object type that has a gadget variant (for example, `XmLabelGadget`) as an attribute of an object declaration. The `ObjectType` can be any object type, including gadgets. You need to specify the `GADGET` or `WIDGET` keyword only in the declaration of an object, not when you reference the object. You cannot specify the `GADGET` or `WIDGET` keyword for a user-defined object; user-defined objects are always widgets.
Identifier Sections
The identifier section allows you to define an identifier, a mechanism that achieves run-time binding of
data to names that appear in a UIL module. The identifier section consists of the reserved IDENTIFIER
keyword, followed by a list of names. Each name is followed by a semicolon (;). The syntax is as follows:

```
IDENTIFIER IdentifierName; [...] 
```

You can use these names later in the UIL module as either the value of an argument to a widget or the
tag value to a callback procedure. At run time, use the MrmRegisterNames and
MrmRegisterNamesInHierarchy MRM functions to bind the identifier name with the data (or, in the case
of callbacks, with the address of the data) associated with the identifier.

Each UIL module has a single name space; therefore, you cannot use the name you used for a value,
object, or procedure as an identifier name in the same module.

The UIL compiler does not do any type checking on the use of identifiers in a UIL module. Unlike a UIL
value, an identifier does not have a UIL type associated with it. Regardless of what particular type a
widget argument or callback procedure tag is defined to be, you can use an identifier in that context
instead of a value of the corresponding type.

To reference these identifier names in a UIL module, use the name of the identifier wherever you want its
value to be used.

Include Directives
The include directive incorporates the contents of a specified file into a UIL module. This mechanism
allows several UIL modules to share common definitions. The syntax for the include directive is as follows:

```
INCLUDE FILE FileName ; 
```

The UIL compiler replaces the include directive with the contents of the include file and processes it as if
these contents were displayed in the current UIL source file.

You can nest include files, meaning that an include file can contain include directives. The UIL compiler
can process up to 100 references (including the file containing the UIL module). Therefore, you can
include up to 99 files in a single UIL module, including nested files. Each time a file is opened counts as a
reference; therefore, including the same file twice counts as two references.

The character expression is a file specification that identifies the file to be included. The rules for finding
the specified file are similar to the rules for finding header, or .h, files using the include directive, #include,
with a quoted string in C language. The uil command uses the -I option for specifying a search directory
for include files. Search rules are as follows:
• If you supply a directory, the UIL compiler searches only that directory for the include file.
• If you do not supply a directory, the UIL compiler searches for the include file in the directory of the
  main source file.
• If the include file is not found in the main source file directory, the compiler looks in the same directory
  as the source file.

Language Syntax
This section contains information on the following:
• Names and Strings
• Data Types
• String Literals
• Integer Literals
Names and Strings
Names can consist of any of the characters A to Z, a to z, 0 to 9, $ (dollar sign), and _ (underscore). Names cannot begin with a digit (0 to 9). The maximum length of a name is 31 characters.

UIL gives you a choice of either case-sensitive or case-insensitive names through a clause in the MODULE header. For example, if names are case-sensitive, the names "sample" and "Sample" are distinct from each other. If names are case-insensitive, these names are treated as the same name and can be used interchangeably. By default, UIL assumes names are case-sensitive.

In case-insensitive mode, the compiler outputs all names in the UID file in uppercase form. In case-sensitive mode, names are displayed in the UIL file exactly as they are displayed in the source file.

The following lists the reserved keywords, which cannot be used for programmer-defined names:

<table>
<thead>
<tr>
<th>ARGUMENTS</th>
<th>CALLBACKS</th>
<th>CONTROLS</th>
</tr>
</thead>
<tbody>
<tr>
<td>END</td>
<td>EXPORTED</td>
<td>FALSE</td>
</tr>
<tr>
<td>GADGET</td>
<td>IDENTIFIER</td>
<td>INCLUDE</td>
</tr>
<tr>
<td>LIST</td>
<td>MODULE</td>
<td>OFF</td>
</tr>
<tr>
<td>ON</td>
<td>OBJECT</td>
<td>PRIVATE</td>
</tr>
<tr>
<td>PROCEDURE</td>
<td>PROCEDURES</td>
<td>TRUE</td>
</tr>
<tr>
<td>VALUE</td>
<td>WIDGET</td>
<td></td>
</tr>
</tbody>
</table>

The following lists UIL unreserved keywords. These keywords can be used as programmer-defined names; however, if you use any of these keywords as names, you cannot use the UIL-supplied form of that keyword.

Built-in argument names (for example, XmNx, XmNheight)
Built-in reason names (for example, XmNactivateCallback, XmNhelpCallback)
Character set names (for example, ISO_LATIN1, ISO_HEBREW_LR)
Constant value names (for example, XmMENU_OPTION, XmBROWSE_SELECT)
Object types (for example, XmPushButton, XmBulletinBoard)
String literals can be composed of uppercase and lowercase letters, digits, and punctuation characters. Spaces, tabs, and comments are special elements in the language. They are a means of delimiting other elements, such as two names. One or more of these elements can be displayed before or after any other element in the language. However, spaces, tabs, and comments that are displayed in string literals are treated as character sequences rather than delimiters.

### Data Types

UIL provides literals for several of the value types it supports. Some of the value types are not supported as literals (for example, pixmaps and string tables). You can specify values for these types by using functions described in the "Functions" section. UIL directly supports the following literal types:

- String literal
- Integer literal
- Boolean literal
- Floating-point literal

UIL also includes the ANY data type, which is used to turn off compile-time checking of data types.

### String Literals

A string literal is a sequence of 0 or more 8-bit or 16-bit characters or a combination delimited by ' (single quotation marks) or " (double quotation marks). String literals can also contain multibyte characters delimited with double quotation marks. String literals can be no more than 2,000 characters long.

A single-quoted string literal can span multiple source lines. To continue a single-quoted string literal, end the continued line with a \ (backslash). The literal continues with the first character on the next line.

Double-quoted string literals cannot span multiple source lines. (Because double-quoted strings can contain escape sequences and other special characters, you cannot use the backslash character to designate the continuation of the string.) To build a string value that must span multiple source lines, use the concatenation operation that is described later in this section.

The syntax of a string literal can be one of the following:

'[[CharacterString]]'

"[#CharSet][[CharacterString]]"

Both string forms associate a character set with a string value. UIL uses the following rules to determine the character set and storage format for string literals:

- A string declared as 'String' is equivalent to "CurCharSet String", where CurCharSet is the codeset portion of the value of the LANG environment variable. If the LANG environment variable is not set or has no code set component, CurCharSet is the value of XmFALLBACK_CHARSET. By default, XmFALLBACK_CHARSET is ISO8859-1 (equivalent to ISO_LATIN1), but vendors can define a different default.
- A string declared as "String" is equivalent to #CharSet String if you specified CharSet as the default character set for the module. If no default character set has been specified for the module and either

<table>
<thead>
<tr>
<th>COLOR</th>
<th>KEYSYM</th>
<th>VERSION</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLOR_TABLE</td>
<td>MANAGED</td>
<td>WIDE_CHARACTER</td>
</tr>
<tr>
<td>COMPOUND_STRING</td>
<td>NAMES</td>
<td>WIDGET</td>
</tr>
<tr>
<td>COMPOUND_STRING_TABLE</td>
<td>OBJECTS</td>
<td>XBITMAPFILE</td>
</tr>
<tr>
<td>RIGHT_TO_LEFT</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
the -s option is provided to the uil command or the use_setlocale_flag value is set for the Uil function callable compiler, the string is interpreted to be a string in the current locale. This means that the string is parsed in the locale of the user by calling setlocale and its character set is set to a value of XmFONTLIST_DEFAULT_TAG. If the string is converted to a compound string, it is stored as a locale-encoded text segment. Otherwise, "String" is equivalent to #CurCharSet"String", where CurCharSet is interpreted as described for single-quoted strings.

- A string of the form "String" or #CharSet"String" is stored as a null-terminated string.

The following lists the character sets supported by the UIL compiler for string literals. Note that several UIL names map to the same character set. In some cases, the UIL name influences how string literals are read. For example, strings identified by a UIL character set name ending in _LR are read left-to-right. Names that end in a different number reflect different fonts (for example, ISO_LATIN1 or ISO_LATIN6). All character sets in this list are represented by 8 bits.

<table>
<thead>
<tr>
<th>UIL Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISO_Latin1</td>
<td>GL: ASCII, GR: Latin-1 Supplement</td>
</tr>
<tr>
<td>ISO_Latin2</td>
<td>GL: ASCII, GR: Latin-2 Supplement</td>
</tr>
<tr>
<td>ISO_Arabic</td>
<td>GL: ASCII, GR: Latin-Arabic Supplement</td>
</tr>
<tr>
<td>ISO_Latin6</td>
<td>GL: ASCII, GR: Latin-Arabic Supplement</td>
</tr>
<tr>
<td>ISO_Greek</td>
<td>GL: ASCII, GR: Latin-Greek Supplement</td>
</tr>
<tr>
<td>ISO_Latin7</td>
<td>GL: ASCII, GR: Latin-Greek Supplement</td>
</tr>
<tr>
<td>ISO_Hebrew</td>
<td>GL: ASCII, GR: Latin-Hebrew Supplement</td>
</tr>
<tr>
<td>ISO_Latin8</td>
<td>GL: ASCII, GR: Latin-Hebrew Supplement</td>
</tr>
<tr>
<td>ISO_Hebrew_LR</td>
<td>GL: ASCII, GR: Latin-Hebrew Supplement</td>
</tr>
<tr>
<td>ISO_Latin8_LR</td>
<td>GL: ASCII, GR: Latin-Hebrew Supplement</td>
</tr>
<tr>
<td>JIS_Katakana</td>
<td>GL: JIS Roman, GR: JIS Katakana</td>
</tr>
</tbody>
</table>

Following are the parsing rules for each of the character sets:

- **Character Set**: All character sets
  - **Parsing Rule**: Character codes in the range 00 to 1F, 7F, and 80 to 9F are control characters including both bytes of 16-bit characters. The compiler flags these as illegal characters. These sets are parsed from left to right. The escape sequences for null-terminated strings are also supported by these character sets.

- **Character Set**: ISO_Latin1, ISO_Latin2, ISO_Arabic, ISO_Latin6, ISO_Greek, ISO_Latin7
  - **Parsing Rule**: These sets are parsed from right to left. For example, the string #ISO_Hebrew"012345" generates a primitive string "543210" with the character set ISO_Hebrew. A DDIS descriptor for such a string has this segment marked as being right to left. The escape sequences for null-terminated strings are also supported by these character sets, and the characters that compose the escape sequences are in left-to-right order. For example, you type \\n, not \n. These sets are parsed from left to right. For example, the string #ISO_Hebrew"012345" generates a primitive string "012345" with the character set ISO_Hebrew. A DDIS descriptor for such a string marks this segment as being left to right. The escape sequences for null-terminated strings are also supported by these character sets. The escape sequences for null-terminated strings are also supported by these character sets.

- **Character Set**: ISO_Hebrew_LR, ISO_Latin8_LR
  - **Parsing Rule**: These sets are parsed from left to right. For example, the string #ISO_Hebrew_LR"012345" generates a primitive string "012345" with the character set ISO_Hebrew. A DDIS descriptor for such a string marks this segment as being right to left. The escape sequences for null-terminated strings are also supported by these character sets. The escape sequences for null-terminated strings are also supported by these character sets. Note that the \ (backslash) can be displayed as a yen symbol.
In addition to designating parsing rules for strings, character set information remains an attribute of a compound string. If the string is included in a string consisting of several concatenated segments, the character set information is included with that string segment. This gives AIXwindows the information it needs to decipher the compound string and choose a font to display the string.

For an application interface displayed only in English, UIL lets you ignore the distinctions between the two uses of strings. The compiler recognizes by context when a string must be passed as a null-terminated string or as a compound string.

The UIL compiler recognizes enough information about the various character sets to correctly parse string literals. The compiler also issues errors if you use a compound string in a context that supports only null-terminated strings.

Since the character set names are keywords, you must put them in lowercase if case-sensitive names are in force. If names are case-insensitive, character set names can be uppercase, lowercase, or mixed case.

In addition to the built-in character sets recognized by UIL, you can define your own character sets with the `CHARACTER_SET` function. You can use the `CHARACTER_SET` function anywhere a character set can be specified.

String literals can contain characters with the eighth (high-order) bit set. You cannot type control characters (00 to 1F, 7F, and 80 to 9F) directly in a single-quoted string literal. However, you can represent these characters with escape sequences. The following list shows the escape sequences for special characters:

- `\b` Backspace
- `\f` Form-feed
- `\n` New-line
- `\r` Carriage return
- `\t` Horizontal tab
- `\v` Vertical tab
- `\'` Single quotation mark
- `\"` Double quotation mark
- `\` Backslash
- `\\` Character whose internal representation is given by Integer (in the range 0 to 255 decimal).

**Note:** Escape sequences are processed literally in strings that are parsed in the current locale (localized strings).

The UIL compiler does not process new-line characters in compound strings. The effect of a new-line character in a compound string depends only on the character set of the string. The result is not guaranteed to be a multiline string.

**Compound String Literals:**

A compound string consists of a string of 8-bit, 16-bit, or multibyte characters, a named character set, and a writing direction. Its UIL data type is `compound_string`.

The writing direction of a compound string is implied by the character set specified for the string. You can explicitly set the writing direction for a compound string by using the `COMPOUND_STRING` function.

A compound string can consist of a sequence of concatenated compound strings, null-terminated strings, or a combination of both, each of which can have a different character set property and writing direction. Use the `&` (ampersand) concatenation operator to create a sequence of compound strings.

Each string in the sequence is stored, including the character set and writing direction information.
Generally, a string literal is stored in the UID file as a compound string when the literal consists of concatenated strings having different character sets or writing directions, or when you use the string to specify a value for an argument that requires a compound string value. If you want to guarantee that a string literal is stored as a compound string, you must use the `COMPOUND_STRING` function.

**Data Storage Consumption for String Literals:**

The way a string literal is stored in the UID file depends on how you declare and use the string. The UIL compiler automatically converts a null-terminated string to a compound string if you use the string to specify the value of an argument that requires a compound string. However, this conversion is costly in terms of storage consumption.

PRIVATE, EXPORTED, and IMPORTED string literals require storage for a single allocation when the literal is declared; thereafter, storage is required for each reference to the literal. Literals declared in-line require storage for both an allocation and a reference.

The following list summarizes data storage consumption for string literals. The storage requirement for an allocation consists of a fixed portion and a variable portion. The fixed portion of an allocation is roughly the same as the storage requirement for a reference (a few bytes). The storage consumed by the variable portion depends on the size of the literal value (the length of the string). To conserve storage space, avoid making string literal declarations that result in an allocation per use.

<table>
<thead>
<tr>
<th>Declaration (and Data Type)</th>
<th>Used As</th>
<th>Storage Requirements Per Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>In-line (Null-terminated)</td>
<td>Null-terminated</td>
<td>An allocation and a reference (within the module)</td>
</tr>
<tr>
<td>Private (Null-terminated)</td>
<td>Null-terminated</td>
<td>A reference (within the module)</td>
</tr>
<tr>
<td>Exported (Null-terminated)</td>
<td>Null-terminated</td>
<td>A reference (within the UID hierarchy)</td>
</tr>
<tr>
<td>Imported (Null-terminated)</td>
<td>Null-terminated</td>
<td>A reference (within the UID hierarchy)</td>
</tr>
<tr>
<td>In-line (Null-terminated)</td>
<td>Compound</td>
<td>An allocation and a reference (within the module)</td>
</tr>
<tr>
<td>Private (Null-terminated)</td>
<td>Compound</td>
<td>An allocation and a reference (within the module)</td>
</tr>
<tr>
<td>Exported (Null-terminated)</td>
<td>Compound</td>
<td>A reference (within the UID hierarchy)</td>
</tr>
<tr>
<td>Imported (Null-terminated)</td>
<td>Compound</td>
<td>A reference (within the UID hierarchy)</td>
</tr>
<tr>
<td>In-line (Compound)</td>
<td>Compound</td>
<td>An allocation and a reference (within the module)</td>
</tr>
<tr>
<td>Private (Compound)</td>
<td>Compound</td>
<td>A reference (within the module)</td>
</tr>
<tr>
<td>Exported (Compound)</td>
<td>Compound</td>
<td>A reference (within the UID hierarchy)</td>
</tr>
<tr>
<td>Imported (Compound)</td>
<td>Compound</td>
<td>A reference (within the UID hierarchy)</td>
</tr>
</tbody>
</table>

**Integer Literals**

An integer literal represents the value of a whole number. Integer literals have the form of an optional sign followed by one or more decimal digits. An integer literal must not contain embedded spaces or commas.

Integer literals are stored in the UID file as long integers. Exported and imported integer literals require a single allocation when the literal is declared; thereafter, a few bytes of storage are required for each reference to the literal. Private integer literals and those declared in-line require allocation and reference storage per use. To conserve storage space, avoid making integer literal declarations that result in an allocation per use.

The following list shows data storage consumption for integer literals:

<table>
<thead>
<tr>
<th>Declaration</th>
<th>Storage Requirements Per Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>In-line</td>
<td>An allocation and a reference (within the module).</td>
</tr>
<tr>
<td>Private</td>
<td>An allocation and a reference (within the module).</td>
</tr>
</tbody>
</table>
Declarations Declaration Storage Requirements Per Use
Exported A reference (within the UID hierarchy)
Imported A reference (within the UID hierarchy)

**Boolean Literals**

A Boolean literal represents the True value (reserved keyword TRUE or On) or False value (reserved keyword FALSE or Off). These keywords are subject to case-sensitivity rules.

In a UID file, TRUE is represented by the integer value 1 and FALSE is represented by the integer value 0.

Data storage consumption for Boolean literals is the same as that for integer literals.

**Floating-Point Literals**

A floating-point literal represents the value of a real (or float) number. Floating-point literals have the following form:

\[
[+|-] [\text{Integer}].\text{Integer}[E|e[+|-]\text{Exponent}]
\]

For maximum portability, a floating-point literal can represent values in the range 1.0E-37 to 1.0E+37 with at least six significant digits. On many machines, this range is wider, with more significant digits. A floating-point literal must not contain embedded spaces or commas.

Floating-point literals are stored in the UID file as double-precision, floating-point numbers. The following gives examples of valid and invalid floating-point notation for the UIL compiler:

<table>
<thead>
<tr>
<th>Valid Floating-Point Literals</th>
<th>Invalid Floating-Point Literals</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>1e1 (no decimal point)</td>
</tr>
<tr>
<td>.1</td>
<td>E-1 (no decimal point or digits)</td>
</tr>
<tr>
<td>3.1415E-2 (equals .031415)</td>
<td>2.87 e6 (embedded blanks)</td>
</tr>
<tr>
<td>-6.29e7 (equals -62900000)</td>
<td>2.0e100 (out of range)</td>
</tr>
</tbody>
</table>

Data storage consumption for floating-point literals is the same as that for integer literals.

**ANY Data Type**

The purpose of the ANY data type is to shut off the data-type checking feature of the UIL compiler. You can use the ANY data type for either of the following:

- Specifying the type of a callback procedure tag.
- Specifying the type of a user-defined argument.

You can use the ANY data type when you need to use a type not supported by the UIL compiler or when you want the data-type restrictions imposed by the compiler to be relaxed. For example, you might want to define a widget having an argument that can accept different types of values, depending on run-time circumstances.

If you specify that an argument takes an ANY value, the compiler does not check the type of the value specified for that argument. Therefore, you need to take care when specifying a value for an argument of the ANY data type. You may get unexpected results at run time if you pass a value having a data type that the widget does not support for that argument.

**Expressions**

UIL includes compile-time value expressions. These expressions can contain references to other UIL values, but cannot be forward-referenced.
The following lists the set of operators in UIL that allow you to create integer, real, and Boolean values based on other values defined with the UIL module. In the list, a precedence of 1 is the highest.

<table>
<thead>
<tr>
<th>Operator (And Its Meaning)</th>
<th>Operand Types</th>
<th>Precedence</th>
</tr>
</thead>
<tbody>
<tr>
<td>~ (NOT)</td>
<td>Boolean</td>
<td>1</td>
</tr>
<tr>
<td>(Ones complement)</td>
<td>integer</td>
<td></td>
</tr>
<tr>
<td>- (Negate)</td>
<td>float</td>
<td>1</td>
</tr>
<tr>
<td>(Negate)</td>
<td>integer</td>
<td></td>
</tr>
<tr>
<td>+ (NOP)</td>
<td>float</td>
<td>1</td>
</tr>
<tr>
<td>(NOP)</td>
<td>integer</td>
<td></td>
</tr>
<tr>
<td>* (Multiply)</td>
<td>float,float</td>
<td>2</td>
</tr>
<tr>
<td>(Multiply)</td>
<td>integer,integer</td>
<td></td>
</tr>
<tr>
<td>/ (Divide)</td>
<td>float,float</td>
<td>2</td>
</tr>
<tr>
<td>(Divide)</td>
<td>integer,integer</td>
<td></td>
</tr>
<tr>
<td>+ (Add)</td>
<td>float,float</td>
<td>3</td>
</tr>
<tr>
<td>(Add)</td>
<td>integer,integer</td>
<td></td>
</tr>
<tr>
<td>- (Subtract)</td>
<td>float,float</td>
<td>3</td>
</tr>
<tr>
<td>(Subtract)</td>
<td>integer,integer</td>
<td></td>
</tr>
<tr>
<td>&gt;&gt; (Shift right)</td>
<td>integer,integer</td>
<td></td>
</tr>
<tr>
<td>&lt;&lt; (Shift left)</td>
<td>integer,integer</td>
<td></td>
</tr>
<tr>
<td>&amp; (NOT)</td>
<td>Boolean,Boolean</td>
<td></td>
</tr>
<tr>
<td>(Bitwise AND)</td>
<td>integer,integer</td>
<td></td>
</tr>
<tr>
<td>(Concatenate)</td>
<td>string,string</td>
<td></td>
</tr>
<tr>
<td>I (OR)</td>
<td>Boolean,Boolean</td>
<td></td>
</tr>
<tr>
<td>(Bitwise OR)</td>
<td>integer,integer</td>
<td></td>
</tr>
<tr>
<td>^ (XOR)</td>
<td>Boolean,Boolean</td>
<td></td>
</tr>
<tr>
<td>(Bitwise XOR)</td>
<td>integer,integer</td>
<td></td>
</tr>
</tbody>
</table>

A string can be either a single compound string or a sequence of compound strings. If the two concatenated strings have different properties (such as writing direction or character set), the result of the concatenation is a multisegment compound string.

The string resulting from the concatenation is a null-terminated string unless one or more of the following conditions exists:
- One of the operands is a compound string.
- The operands have different character set properties.
- The operands have different writing directions.

If one or more of previous conditions are met, the resulting string is a compound string. You cannot use imported or exported values as operands of the concatenation operator.

The result of each operator has the same type as its operands. You cannot mix types in an expression without using conversion functions.

You can use parentheses to override the normal precedence of operators. In a sequence of unary operators, the operations are performed in right-to-left order. For example, - + -A is equivalent to
- (+(-A)). In a sequence of binary operators of the same precedence, the operations are performed in left-to-right order. For example, \(A*B/C*D\) is equivalent to \(((A*B)/c)*D\).

A value declaration gives a value a name. You cannot redefine the value of that name in a subsequent value declaration. You can use a value containing operators and functions anywhere you can use a value in a UIL module. You cannot use imported values as operands in expressions.

Several of the binary operators are defined for multiple data types. For example, the operator for multiplication (\(\cdot\)) is defined for both floating-point and integer operands.

For the UIL compiler to perform these binary operations, both operands must be of the same type. If you supply operands of different data types, the UIL compiler automatically converts one of the operands to the type of the other according to the following conversion rules:

- If the operands are an integer and a Boolean, the Boolean is converted to an integer.
- If the operands are an integer and a floating-point, the integer is converted to a floating-point.
- If the operands are a floating-point and a Boolean, the Boolean is converted to a floating-point.

You can also explicitly convert the data type of a value by using one of the INTEGER, FLOAT, or SINGLE_FLOAT conversion functions.

Functions
UIL provides functions to generate the following types of values:

- Character sets
- Keysyms
- Colors
- Pixmaps
- Single-precision, floating-point numbers
- Double-precision, floating-point numbers
- Fonts
- Font sets
- Font tables
- Compound strings
- Compound string tables
- ASCIZ (null-terminated) string tables
- Wide character strings
- Widget class names
- Integer tables
- Arguments
- Reasons
- Translation tables.

All examples in the following sections assume case-insensitive mode. Keywords are shown in uppercase letters to distinguish them from user-specified names, which are shown in mixed-case italics. This use of uppercase letters is not required in case-insensitive mode. In case-sensitive mode, keywords must be in lowercase letters.

\[\text{CHARACTER\_SET}(\text{StringExpression}[\text{Property}[\ldots]])\]

You can define your own character sets with the CHARACTER\_SET function. You can use the CHARACTER\_SET function anywhere a character set can be specified.

The result of the CHARACTER\_SET function is a character set with the name \text{StringExpression} and the properties you specify. \text{StringExpression} must be a null-terminated string. You can
optionally include one or both of the following clauses to specify properties for the resulting character set: \texttt{RIGHT\_TO\_LEFT} = \texttt{BooleanExpression} \texttt{SIXTEEN\_BIT} = \texttt{BooleanExpression}

The \texttt{RIGHT\_TO\_LEFT} clause sets the default writing direction of the string from right to left if \texttt{BooleanExpression} is True, and left to right otherwise.

The \texttt{SIXTEEN\_BIT} clause allows the strings associated with this character set to be interpreted as 16-bit characters if \texttt{BooleanExpression} is True, and 8-bit characters otherwise.

\texttt{KEYSYM(StringLiteral)}

The \texttt{KEYSYM} function is used to specify a keysym for a mnemonic resource. The \texttt{StringLiteral} must contain exactly one character. If the -s compiler flag is used, \texttt{StringLiteral} which uses double quotes must specify a character set.

\texttt{COLOR(StringExpression[,\texttt{FOREGROUND}|\texttt{BACKGROUND}]})

The \texttt{COLOR} function supports the definition of colors. Using the \texttt{COLOR} function, you can designate a value to specify a color and use that value for arguments requiring a color value. The string expression names the color you want to define. The optional \texttt{FOREGROUND} and \texttt{BACKGROUND} keywords identify how the color is to be displayed on a monochrome device when the color is used in the definition of a color table.

The UIL compiler does not have built-in color names. Colors are a server-dependent attribute of an object. Colors are defined on each server and may have different red-green-blue (RGB) values on each server. The string you specify as the color argument must be recognized by the server on which your application runs.

In a UID file, UIL represents a color as a character string. MRM calls X translation functions that convert a color string to the device-specific pixel value. If you are running on a monochrome server, all colors translate to black or white. If you are on a color server, the color names translate to their proper colors if the following conditions are met:

- The color is defined.
- The color map is not yet full.

If the color map is full, even valid colors translate to black or white (foreground or background).

Generally, interfaces do not specify colors for widgets. This enables the selection of colors to be controlled by the user through the .Xdefaults file.

To write an application that runs on both monochrome and color devices, you need to specify which colors in a color table (defined with the \texttt{COLOR\_TABLE} function) map to the background and which colors map to the foreground. UIL lets you use the \texttt{COLOR} function to map the color red to the background color on a monochrome device as follows:

\begin{verbatim}
VALUE c: COLOR ( 'red',BACKGROUND );
\end{verbatim}

Mapping is necessary only when the MRM is given a color and the application is to be displayed on a monochrome device. In this case, each color is considered to be in one of the following three categories:

- The color is mapped to the background color on the monochrome device.
- The color is mapped to the foreground color on the monochrome device.
- Monochrome mapping is undefined for this color.

If the color is mapped to the foreground or background color, MRM substitutes the foreground or background color, respectively. If you do not specify the monochrome mapping for a color, MRM passes the color string to AIXwindows for mapping to the foreground or background color.

\texttt{RGB(RedInteger, GreenInteger, BlueInteger)}

The three integers define the values for the red, green, and blue components of the color, in that order. The values of these components can range from 0 to 65,535, inclusive.

In a UID file, UIL represents an RGB value as three integers. MRM calls X translation functions that convert the integers to the device-specific pixel value. If you are running on a monochrome
server, all colors translate to black or white. If you are on a color server, RGB values translate to
their proper colors if the color map is not yet full. If the color map is full, values translate to black
or white (foreground or background).

COLOR_TABLE(ColorExpression='Character', ...)
The color expression is a previously defined color, a color defined in-line with the COLOR function,

or the phrase BACKGROUND COLOR or FOREGROUND COLOR. The character can be any
valid UIL character.

The COLOR_TABLE function provides a device-independent way to specify a set of colors. The
COLOR_TABLE function accepts either previously defined UIL color names or in-line color
definitions (using the COLOR function). A color table must be private because its contents must be
known by the UIL compiler to construct an icon. The colors within a color table, however, can be
imported, exported, or private.

The single letter associated with each color is the character you use to represent that color when
creating an icon. Each letter used to represent a color must be unique within the color table.

ICON([COLOR_TABLE=ColorTableName,] Row[,...])
The color table name must refer to a previously defined color table. The row is a character
expression that gives one row of the icon.

The ICON function describes a rectangular icon that is $x$ pixels wide and $y$ pixels high. The strings
surrounded by single quotation marks describe the icon. Each string represents a row in the icon;
each character in the string represents a pixel.

The first row in an icon definition determines the width of the icon. All rows must have the same
number of characters as the first row. The height of the icon is dictated by the number of rows.

The first argument of the ICON function (the color table specification) is optional and identifies the
colors that are available in this icon. By using the single letter associated with each color, you can
specify the color of each pixel in the icon. The icon must be constructed of characters defined in

the specified color table.

A default color table is used if you omit the argument specifying the color table. To make use of
the default color table, the rows of your icon must contain only spaces and asterisks. The default
color table is defined as follows:

COLOR_TABLE( BACKGROUND COLOR = ' ', FOREGROUND COLOR = '*' )

You can define other characters to represent the background color and foreground color by
replacing the space and asterisk in the BACKGROUND COLOR and FOREGROUND COLOR
clauses shown in the example statement. You can specify icons as private, imported, or exported.
Use the MrmFetchIconLiteral MRM function to retrieve an exported icon at run time.

XBITMAPFILE(StringExpression)
The XBITMAPFILE function is similar to the ICON function in that both describe a rectangular icon
that is $x$ pixels wide and $y$ pixels high. However, the XBITMAPFILE function allows you to specify
an external file containing the definition of an X bitmap, while all ICON function definitions must be
coded directly within UIL. X bitmap files can be generated by many different X applications. UIL
reads these files through the XBITMAPFILE function, but does not support creation of these files.
The X bitmap file specified as the argument to the XBITMAPFILE function is read by MRM at
application run time.

The XBITMAPFILE function returns a value of type pixmap and can be used anywhere a pixmap
data type is expected.

SINGLE_FLOAT(RealNumberLiteral)
The SINGLE_FLOAT function lets you store floating-point literals in UIL files as single-precision,
floating-point numbers. Single-precision, floating-point numbers can often be stored using less
memory than double-precision, floating-point numbers. The RealNumberLiteral can be either an integer literal or a floating-point literal. A value defined using this function cannot be used in an arithmetic expression.

FLOAT(RealNumberLiteral)
The FLOAT function lets you store floating-point literals in UIL files as double-precision, floating-point numbers. The RealNumberLiteral can be either an integer literal or a floating-point literal.

FONT(StringExpression[,CHARACTER_SET=CharSet])
You define fonts with the FONT function. Using the FONT function, you designate a value to specify a font and use that value for arguments that require a font value. The UIL compiler has no built-in fonts.

Each font makes sense only in the context of a character set. The FONT function has an additional parameter to let you specify the character set for the font. This parameter is optional; if you omit it, the default character set depends on the value of the LANG environment variable. If LANG is not set, the default character set is set to XmFALLBACK_CHARSET.

The string expression specifies the name of the font and the clause CHARACTER_SET=CharSet specifies the character set for the font. The string expression used in the FONT function cannot be a compound string.

FONTSET(StringExpression[,...][,CHARACTER_SET=CharSet])
You define fontsets with the FONTSET function. Using the FONTSET function, you designate a set of values to specify a font and use those values for arguments that require a fontset value. The UIL compiler has no built-in fonts.

Each font makes sense only in the context of a character set. The FONTSET function has an additional parameter to let you specify the character set for the font. This parameter is optional; if you omit it, the default character set depends on the value of the LANG environment variable. If LANG is not set, the default character set is set to XmFALLBACK_CHARSET.

The string expression specifies the name of the font and the clause CHARACTER_SET=CharSet specifies the character set for the font. The string expression used in the FONTSET function cannot be a compound string.

FONT_TABLE(FontExpression[,...])
A font table is a sequence of pairs of fonts and character sets. At run time when an object needs to display a string, the object scans the font table for the character set that matches the character set of the string to be displayed. UIL provides the FONT_TABLE function to let you supply such an argument. The font expression is created with the FONT and FONTSET functions.

If you specify a single font value to specify an argument that requires a font table, the UIL compiler automatically converts a font value to a font table.

COMPOUND_STRING(StringExpression[,.Property[,...]])
Use the COMPOUND_STRING function to set properties of a null-terminated string and to convert it into a compound string. The properties you can set are the character set, writing direction, and separator.

The result of the COMPOUND_STRING function is a compound string with the string expression as its value. You can optionally include one or more of the following clauses to specify properties for the resulting compound string:

CHARACTER_SET=CharacterSet
RIGHT_TO_LEFT=BooleanExpression
SEPARATE=BooleanExpression

The CHARACTER_SET clause specifies the character set for the string. If you omit the CHARACTER_SET clause, the resulting string has the same character set as StringExpression.
The **RIGHT_TO_LEFT** clause sets the writing direction of the string from right to left if *BooleanExpression* is True. Otherwise, writing direction is left to right. Specifying this argument does not cause the value of the string expression to change. If you omit the **RIGHT_TO_LEFT** argument, the resulting string has the same writing direction as *StringExpression*.

The **SEPARATE** clause appends a separator to the end of the compound string if *BooleanExpression* is True. If you omit the **SEPARATE** clause, the resulting string does not have a separator.

You cannot use imported or exported values as the operands of the **COMPOUND_STRING** function.

**COMPOUND_STRING_TABLE**(*StringExpression[,...]*)

A compound string table is an array of compound strings. Objects requiring a list of string values, such as the *XmNItems* and *XmNselectedItems* arguments for the **List** widget, use string table values. The **COMPOUND_STRING_TABLE** function builds the values for these two arguments of the **List** widget. The **COMPOUND_STRING_TABLE** function generates a value of type string_table. The name **STRING_TABLE** is a synonym for **COMPOUND_STRING_TABLE**.

The strings inside the string table can be simple strings, which the UIL compiler automatically converts to compound strings.

**ASCIZ_STRING_TABLE**(*StringExpression[,...]*)

An ASCIZ string table is an array of ASCIZ (null-terminated) string values separated by commas. This function allows you to pass more than one ASCIZ string as a callback tag value. The **ASCIZ_STRING_TABLE** function generates a value of type asciz_table. The name **ASCIZ_TABLE** is a synonym for **ASCIZ_STRING_TABLE**.

**WIDE_CHARACTER**(*StringExpression*)

Use the **WIDE_CHARACTER** function to generate a wide character string from a null-terminated string in the current locale.

**CLASS_REC_NAME**(*StringExpression*)

Use the **CLASS_REC_NAME** function to generate a widget class name. For a widget class defined by the toolkit, the string argument is the name of the class. For a user-defined widget, the string argument is the name of the creation function for the widget.

**INTEGER_TABLE**(*IntegerExpression[,...]*)

An integer table is an array of integer values separated by commas. This function allows you to pass more than one integer per callback tag value. The **INTEGER_TABLE** function generates a value of type integer_table.

**ARGUMENTS**(*StringExpression[,ArgumentType]*)

The **ARGUMENTS** function defines the arguments to a user-defined widget. Each of the objects that can be described by UIL permits a set of arguments. For example, *XmNheight* is an argument to most objects and has the integer data type. To specify height for a user-defined widget, you can use the built-in argument name *XmNheight* and specify an integer value when you declare the user-defined widget. Do not use the **ARGUMENTS** function to specify arguments that are built into the UIL compiler.

The *StringExpression* name is the name the UIL compiler uses for the argument in the UID file. The *ArgumentType* is the type of value that can be associated with the argument. If you omit the second argument, the default type is **ANY** and no value type checking occurs. Use any of the following keywords to specify the argument type:

- Any
- Asciz_Table
- Boolean
- Color
- Color_Table
You can use the ARGUMENTS function to allow the UIL compiler to recognize extensions to AIXwindows. For example, an existing widget can accept a new argument. Using the ARGUMENTS function, you can make this new argument available to the UIL compiler before the updated version of the compiler is released.

**REASON(StringExpression)**

The REASON function is useful for defining new reasons for user-defined widgets.

Each of the objects in AIXwindows defines a set of conditions under which it calls a user-defined function. These conditions are known as callback reasons. The user-defined functions are called callback procedures. In a UIL module, you use a callbacks list to specify which user-defined functions are to be called for which reasons.

When you declare a user-defined widget, you can define callback reasons for that widget using the REASON function. The string expression specifies the argument name stored in the UID file for the reason. This reason name is supplied to the widget creation function at run time.

**TRANSLATION_TABLE(StringExpression[,....])**

Each of the AIXwindows widgets have a translation table that maps X events (for example, pressing mouse button 1) to a sequence of actions. Through widget arguments, such as the common translations argument, you can specify an alternate set of events or actions for a particular widget. The TRANSLATION_TABLE function creates a translation table that can be used as the value of an argument that is of the data type translation_table.

You can use one of the following translation table directives with the TRANSLATION_TABLE function: #override, #augment, or #replace. The default is #replace. If you specify one of these directives, it must be the first entry in the translation table.

The #override directive causes any duplicate translations to be ignored. For example, if a translation for <Btn1Down> is already defined in the current translations for a PushButton, the translation defined by NewTranslations overrides the current definition. If the #augment directive is specified, the current definition takes precedence. The #replace directive replaces all current translations with those specified in the XmNtranslations resource.

**Files**

/usr/include/uil/Uil.h
/usr/include/uil/UIlDBDef.h
/usr/include/uil/UIlDef.h
/usr/include/uil/UIlSymDef.h
utmp, wtmp, failedlogin File Format

Purpose
Describes formats for user and accounting information.

Description
The utmp file, the wtmp file, and the failedlogin file contain records with user and accounting information.

When a user attempts to log in, the login program writes entries in two files:

- The /etc/utmp file, which contains a record of users logged into the system.
- The /var/adm/wtmp file (if it exists), which contains connect-time accounting records.

On an invalid login attempt, due to an incorrect login name or password, the login program makes an entry in:

- The /etc/security/failedlogin file, which contains a record of unsuccessful login attempts.

The records in these files follow the utmp format, defined in the utmp.h header file.

Files
/etc/utmp	Contains a record of users logged into the system.
/var/adm/wtmp	Contains connect accounting information.
/etc/security/failedlogin	Contains a record of invalid login attempts.

Related Information
The fwtmp command, init command, login command, su command, who command.

The utmp.h file, lastlog file format.

Accounting commands in Operating system and device management lists accounting commands that run automatically or keyboard commands entered from the keyboard.

System accounting in Operating system and device management and Setting up an accounting subsystem in Operating system and device management.

vgrindefs File Format

Purpose
Contains the language definition database for the vgrind command.
Description
The **vgrindefs** file format contains all the language definitions for the **vgrind** command. The database is very similar to the **terminfo** file format (file of terminal capabilities).

Fields
The following table contains the name and description of each field:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ab</td>
<td>str</td>
<td>Alternate regular expression for the start of a comment.</td>
</tr>
<tr>
<td>ae</td>
<td>str</td>
<td>Alternate regular expression for the end of a comment.</td>
</tr>
<tr>
<td>pb</td>
<td>str</td>
<td>Regular expression for the start of a procedure.</td>
</tr>
<tr>
<td>bb</td>
<td>str</td>
<td>Regular expression for the start of a lexical block.</td>
</tr>
<tr>
<td>be</td>
<td>str</td>
<td>Regular expression for the end of a lexical block.</td>
</tr>
<tr>
<td>cb</td>
<td>str</td>
<td>Regular expression for the start of a comment.</td>
</tr>
<tr>
<td>ce</td>
<td>str</td>
<td>Regular expression for the end of a comment.</td>
</tr>
<tr>
<td>sb</td>
<td>str</td>
<td>Regular expression for the start of a string.</td>
</tr>
<tr>
<td>se</td>
<td>str</td>
<td>Regular expression for the end of a string.</td>
</tr>
<tr>
<td>lb</td>
<td>str</td>
<td>Regular expression for the start of a character constant.</td>
</tr>
<tr>
<td>le</td>
<td>str</td>
<td>Regular expression for the end of a character constant.</td>
</tr>
<tr>
<td>tl</td>
<td>bool</td>
<td>Presence means procedures are only defined at the top lexical level.</td>
</tr>
<tr>
<td>oc</td>
<td>bool</td>
<td>Presence means upper and lowercase are equivalent.</td>
</tr>
<tr>
<td>kw</td>
<td>str</td>
<td>List of keywords separated by spaces.</td>
</tr>
</tbody>
</table>

Examples
The following entry, which describes the C language, is typical of a language entry:

```
C|c: :pb="\d?*\d?\p\d??):bb={:be=}:cb=/*:ce=*/:sb=":se=":
 :lb=':le=':tl: :kw=asm
auto break case char continue default do double else enum extern float fortran goto if int long register return short sizeof static struct switch typedef union unsigned while #define
 #else endif #ifndef ifndef include undef # define else endif
 if ifdef ifndef include undef:
```

The first field is the language name or any variants of the name. Thus the C language can be specified to the **vgrind** command in either lowercase or uppercase c.

Entries can continue onto multiple lines by giving a \ (backslash) as the last character of a line. The **vgrindefs** file format has the following two capabilities:

- Boolean capabilities that indicate a particular feature of the language
- String capabilities that give a regular expression or keyword list.

In Java™, where comments can be delimited either by a starting "/\**" or an ending "**/", or by a starting "/\" and "end" at the end of the line, the Java **vgrindefs** definition might be:

```
cb=/*:ce==/:ab=//:ae=$
```
Regular Expressions
The vgrindefs file format uses regular expressions similar to those of the ex command and the lex command. The characters ^ (caret), $ (dollar sign), : (colon), and \ (backslash) are reserved characters and must be quoted with a preceding \ (backslash) if they are to be included as normal characters. The metasymbols and their meanings follow:

- End of a line.
- Beginning of a line.
- Delimiter (space, tab, newline, start of line).
- Matches any string of symbols, such as .* in the lex command.
- Matches any alphanumeric name. In a procedure definition (pb), the string that matches this symbol is used as the procedure name.
- Last item is optional.
- Preceding any string, means that the string does not match an input string if the input string is preceded by an escape character (\). Typically used for languages (such as C) that can include the string delimiter in a string by escaping it.

Unlike other regular expressions in the system, these metasymbols match words and not characters. Hence the pattern "(tramp|steamer)flies?" matches "tramp," "steamer," "trampflies," or "steamerflies."

Keyword List
The keyword list lists keywords in the language, separated by spaces. If the oc field is specified, indicating that uppercase and lowercase are equivalent, then all the keywords should be specified in lowercase.

Files
/usr/share/lib/vgrindefs Contains terminal descriptions.

Related Information
The ex command, lex command, troff command, vgrind command.

The terminfo file format.

WML File Format

Purpose
Generates variable UIL compiler components.

Description
The widget meta-language facility (WML) is used to generate changeable components of the user interface language (UIL) compiler, depending on the widget set. Using WML, you can add new widget UIL support to the AIXwindows widget set or add support for a totally new widget set.

File Format
WML files are ASCII files and can be modified with any standard text editor. They are accessed by WML in the tools/wml directory and have a .wml suffix. The Motif AIXwindows widget set is described in the motif.wml file. This is also the default WML file when using the WML facility.

When creating a WML file to add new widgets or change widget characteristics, you can make a copy of the motif.wml file and modify it. If you are creating a new widget set for use with UIL, create a completely
new file. In either case, the motif.wml file is a good example of WML syntax and can help familiarize you with the language before attempting to write your own WML file.

WML files have a basic syntax that is similar in structure to UIL. WML syntax is made up of the following elements:

- Comments
- Data Type Definitions
- Character Set Definitions
- Enumeration Set Definitions
- Control List Definitions
- Class Definitions
- Child Definitions
- Resource Definitions

You can use spaces, tabs, or new-line characters anywhere in syntax, as long as they do not split keywords or strings. Comments end at a new-line character. The order of elements in syntax is not important.

The widget meta-language syntax examples shown use the following additional conventions:

- [ ] Indicates optional elements.
- ... Indicates where an element of syntax can be repeated.
- | Indicates a choice among multiple items.

### Comments

You can include comments in the WML file. Comments have the following syntax:

```
[AnyElement]!AnyComment
```

Comments begin with an ! (exclamation point) and extend to the end of the line. A comment can begin on a line by itself or follow any part of another element. A comment does not change the meaning of any other element. For example:

```
!This is a comment
! that spans two lines.
DataType !This is a comment that follows code.
```

### Data Type Definitions

*Data type definitions* register all the resource data types used in the file. You must register all the data types used in your WML file. Data type definitions have the following syntax:

```
DataType AnyDatatype [{ InternalLiteral = InternalName | DocName = "String"; [...] }];
```

A data type definition begins with the `DataType` keyword. Following the `DataType` keyword is a list of data types that can be modified with the following:

- `InternalLiteral` Forces the value of the internal symbol table literal definition of the data type name. This modifier is used only to circumvent symbol table definitions hard-coded into the UIL compiler and should be used sparingly.
DocName gives an arbitrary string for use in the documentation. This string supplies a different name for the data type or a single name for the data type if the data type has aliases.

For example:
```
DataType OddNumber {DocName="OddNumber"};
   NewString;
```

**Character Set Definitions**

*Character set definitions* register the AIXwindows Toolkit name and other information for the character set names used in UIL. Character set definitions have the following syntax:

```
CharacterSet
AnyCharacterSet
{ [ FontListElementTag | XmStringCharsetName ] = "String";
 [ Alias = "String" ... ];
 Direction = [ LeftToRight | RightToLeft ];
 ParseDirection = [ LeftToRight | RightToLeft ];
 CharacterSize = [ OneByte | TwoByte ];
 [ ... ]};
[ ... ]
```

A character set definition begins with the `CharacterSet` keyword. Following the `CharacterSet` keyword is a list of character sets that can be modified with the following:

**FontListElementTag | XmStringCharsetName**

- **Alias**

  Specifies the name of the character set. The set specified becomes the character set component of the compound string segment that is created. One of these character sets must be specified.

- **Direction**

  Specifies the direction of a compound string segment created using this character set. The default is `LeftToRight`.

- **ParseDirection**

  Specifies the direction in which an input string is parsed when a compound string segment is created using this character set. If this is not specified, the value of `Direction` is the default.

- **CharacterSize**

  Specifies the number of bytes in each character of a compound string segment created using this character set. The default is `OneByte`.

An example of the character set definition syntax is as follows:
```
CharacterSet
  iso_latin1
  { XmStringCharsetName = "ISO8859-1";
    Alias = "ISOLatin1"; }
  iso_hebrew_lr
  { XmStringCharsetName = "ISO8859-8";
    Alias = "iso_latin8_lr";
```
Enumeration Set Definitions

*Enumeration set definitions* register the named constants used in the AIXwindows Toolkit to specify certain resource values. Enumeration set definitions have the following syntax:

```plaintext
EnumerationSet
  ResourceName : ResourceType
  { EnumerationValueName ; [ ... ] } ;
```

An enumeration set definition begins with the `EnumerationSet` keyword. For each enumeration set defined, the name and type of the resource is listed. The resource name is the AIXwindows Toolkit resource name, with the beginning `XmN` prefix removed and the initial letter capitalized. For example, the name of the AIXwindows Toolkit resource `XmNrowColumnType` would be `RowColumnType`. The resource type is the data type for the resource; for most resources, this is the integer data type. Following the resource name and type is a list of enumeration value names that can be used as settings for the resource. These names are the same as those in the AIXwindows Toolkit.

An example of the enumeration set definition syntax is as follows:

```plaintext
EnumerationSet
  RowColumnType: integer
  { XmWORK_AREA; XmMENU_BAR; XmMENU_POPUP;
    XmMENU_PULLDOWN; XmMENU_OPTION; } ;
```

Control List Definitions

*Control list definitions* assign a name to groups of controls. You can use these control lists later in class definitions to simplify the structure of your WML file. Control list definitions have the following syntax:

```plaintext
ControlList
  AnyControlList [ [ AnyControl; [ ... ] ] ];
```

A control list definition starts with the `ControlList` keyword. Following the `ControlList` keyword are any number of control list definitions. Control list definitions are made up of a control list name followed by the set of controls it represents. For example:

```plaintext
ControlList
  Buttons { PushButton; 
    RadioButton;
    CascadeButton;
    NewCascadebutton; } ;
```

Each control specified in the control list must be defined as a class in the file.

Class Definitions

*Class definitions* describe a particular widget class. Included in this description is its position in the class hierarchy, toolkit convenience function, resources, and controls. There should be one class definition for each widget or gadget in the widget set you want to support in UIL. Class definitions have the following syntax:

```plaintext
Class ClassName : MetaClass | Widget | Gadget 
  {{{
    SuperClass = ClassName; | 
    ParentClass = ParentClassName; | 
    InternalLiteral = InternalName; |
```
Class definitions start with the Class keyword. For each class defined, the name of the class and whether the class is a metaclass, widget, or gadget is listed. Each class definition can be modified using the following:

SuperClass
Indicates the name of the parent class. Only the root of the hierarchy does not specify a super class.

ParentClass
Indicates the name of the widget’s automatically created parent class, if one exists. This allows resources for the automatically created parent class to be used in this class definition. For example, XmBulletinBoardDialog creates both an XmBulletinBoard and an XmDialogShell. To access the resources of the XmDialogShell parent class, specify it here.

InternalLiteral
Forces the value of the internal symbol table literal definition of the class name. This modifier is used only to circumvent symbol table definitions hard-coded into the UIL compiler and should be used sparingly.

Alias
Indicates alternate class names for use in a UIL specification.

ConvenienceFunction
Indicates the name of the creation convenience function for this class. All widget and gadget classes must have ConvenienceFunction specified.

WidgetClass
Indicates the associated widget class of gadget type classes. This value is currently not recognized.

DocName
Defines an arbitrary string for use in the documentation. This value is currently not recognized.

DialogClass
Indicates whether the class is a dialog class. This value is currently not recognized.

Resources
Lists the resources of the widget class. This keyword can be further modified with the following:

Default
Specifies a new default value for this resource. Resource default values are usually set in the resource definition. If an inherited resource's default value is changed by the class, the new default value should be noted here.

Exclude
Specifies whether an inherited resource should be excluded from the resource list of the class. The default value is False.

Children
Lists the names of the automatically created children of this class. This allows those children to be accessed in the UIL file.

Controls
Lists the controls that the widget class allows. The controls can be other classes or a control list from the control definition list.

An example of the usage of the preceding data type and control list definitions is shown:

```plaintext
Class
TopLevelWidget : MetaClass
{
  Resources
  {
```

Chapter 2. File Formats  683
NewWidget : Widget
{
    SuperClass = TopLevelWidget;
    ConvenienceFunction = XtbCreateNewWidget;
    Resources
    {
        XtbNnewResource;
        XtbNfirstResource
        {Default="XtbNEW_VALUE";};
        XtbNsecondResource
        {Exclude=True;};
    },
    Controls
    {
        NewWidget;
        Buttons;
    },
};

**Child Definitions**
Child definitions register the classes of automatically created children. Automatically created children are referenced elsewhere in a UIL file using the *Children* keyword within a class definition. Child definitions have the following syntax:

```
Child
ChildName : ClassName;
[...]
```

*ChildName* is the name of the automatically created child and *ClassName* is the name of the class of that child.

**Resource Definitions**
Resource definitions describe a particular resource. Included in this description is its type and default value. Each new resource reference in a class definition should have a resource definition. Resource definitions have the following syntax:

```
Resource
ResourceName : Argument | Reason | Constraint | SubResource
[[]]
[[]]
```

Resource definitions start with the *Resource* keyword. For each resource definition, the name of the resource and whether the resource is an argument, reason, constraint, or subresource is listed.

**Argument** Indicates a standard resource.
**Reason** Indicates a callback resource.
Constraint Indicates a constraint resource.
SubResource This value is currently not recognized.

A resource definition can be modified with the following:

Type Indicates the data type of the resource. The data type specified must be listed in the data type definition.

ResourceLiteral Indicates the keyword used in the UIL file to reference the resource. In AIXwindows, the resource name is the same as the resource literal name (ResourceLiteral).

InternalLiteral Forces the value of the internal symbol table literal definition of the resource name. This modifier is used only to circumvent symbol table definitions hard-coded into the UIL compiler and should be used sparingly.

Alias Indicates alternate names for the resources used in a UIL specification.

Related Special purpose field that allows resources that act as a counter for the current resources to be related to the resource. UIL automatically sets the value of this related resource to the number of items in the compiled instance of the ResourceName type.

Default Indicates the default value of the resource.

DocName Defines an arbitrary string for use in the documentation. This value is currently not recognized.

An example of the usage of data type, control list, and class definitions is shown:

Resource
  XtbNfirstResource : Argument
    { Type = OddNumber;
      Default = "XtbOLD_VALUE";};
  XtbNsecondResource : Argument
    { Type = NewString;
      Default = "XtbNEW_STRING";};
  XtbNnewResource : Argument
    { Type = OddNumber;
      Default = "XtbODD_NUMBER";};

Related Information
The UIL file format.

XCOFF Object File Format

Purpose
The extended common object file format (XCOFF) is the object file format for the operating system. XCOFF combines the standard common object file format (COFF) with the TOC module format concept, which provides for dynamic linking and replacement of units within an object file. A variation of XCOFF is used for 64-bit object files and executable files.

XCOFF is the formal definition of machine-image object and executable files. These object files are produced by language processors (assemblers and compilers) and the binder (or link editor), and are used primarily by the binder and the system loader.

The default name for an XCOFF executable file is a.out.

Note: This information lists bits in big-endian order.

Read the following information to learn more about XCOFF object files:
  • Composite File Header
  • Sections and Section Headers
Writing Applications that Use XCOFF Declarations

Programs can be written to understand 32-bit XCOFF files, 64-bit XCOFF files, or both. The programs themselves may be compiled in 32-bit mode or 64-bit mode to create 32-bit or 64-bit programs. By defining preprocessor macros, applications can select the proper structure definitions from the XCOFF header files.

**Note:** This document uses "XCOFF32" and "XCOFF64" as shorthand for "32-bit XCOFF" and "64-bit XCOFF", respectively.

**Selecting XCOFF32 Declarations**

To select the XCOFF32 definitions, an application merely needs to include the appropriate header files. Only XCOFF32 structures, fields, and preprocessor defines will be included.

**Selecting XCOFF64 Declarations**

To select the XCOFF64 definitions, an application should define the preprocessor macro `__XCOFF64__`. When XCOFF header files are included, the structures, fields, and preprocessor defines for XCOFF64 will be included. Where possible, the structure names and field names are identical to the XCOFF32 names, but field sizes and offsets may differ.

**Selecting Both XCOFF32 and XCOFF64 Declarations**

To select structure definitions for both XCOFF32 and XCOFF64, an application should define both the preprocessor macros `__XCOFF32__` and `__XCOFF64__`. This will define structures for both kinds of XCOFF files. Structures and typedef names for XCOFF64 files will have the suffix "_64" added to them. (Consult the header files for details.)

**Selecting Hybrid XCOFF Declarations**

An application may choose to select single structures that contain field definitions for both XCOFF32 and XCOFF64 files. For fields that have the same size and offset in both XCOFF32 and XCOFF64 definitions, the field names are retained. For fields whose size or offset differ between XCOFF32 and XCOFF64 definitions, the XCOFF32 fields have a "32" suffix, while the XCOFF64 fields have a "64" suffix. To select hybrid structure definitions, an application should define the preprocessor macro `__XCOFF_HYBRID__`. For example, the symbol table definition (in `/usr/include/syms.h`) will have the name `n_offset32` used for the `n_offset` field for XCOFF32, and the name `n_offset64` used for the `n_offset` field for XCOFF64.

**Understanding XCOFF**

Assemblers and compilers produce XCOFF object files as output. The binder combines individual object files into an XCOFF executable file. The system loader reads an XCOFF executable file to create an executable memory image of a program. The symbolic debugger reads an XCOFF executable file to provide symbolic access to functions and variables of an executable memory image.

An XCOFF file contains the following parts:

- A composite header consisting of:
  - A file header
  - An optional auxiliary header
  - Section headers, one for each of the file's raw-data sections
- Raw-data sections, at most one per section header
- Optional relocation information for individual raw-data sections
- Optional line number information for individual raw-data sections
• An optional symbol table
• An optional string table, which is used for all symbol names in XCOFF64 and for symbol names longer than 8 bytes in XCOFF32.

Not every XCOFF file contains every part. A minimal XCOFF file contains only the file header.

Object and Executable Files
XCOFF object files and executable files are similar in structure. An XCOFF executable file (or "module") must contain an auxiliary header, a loader section header, and a loader section.

The loader raw-data section contains information needed to dynamically load a module into memory for execution. Loading an XCOFF executable file into memory creates the following logical segments:
• A text segment (initialized from the .text section of the XCOFF file).
• A data segment, consisting of initialized data (initialized from the .data section of the XCOFF file) followed by uninitialized data (initialized to 0). The length of uninitialized data is specified in the .bss section header of the XCOFF file.

The XCOFF file Organization illustrates the structure of the XCOFF object file.

XCOFF Header Files
The xcoff.h file defines the structure of the XCOFF file. The xcoff.h file includes the following files:

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filehdr.h</td>
<td>Defines the file header.</td>
</tr>
<tr>
<td>aouthdr.h</td>
<td>Defines the auxiliary header.</td>
</tr>
<tr>
<td>scnhdr.h</td>
<td>Defines the section headers.</td>
</tr>
<tr>
<td>loader.h</td>
<td>Defines the format of raw data in the .loader section.</td>
</tr>
<tr>
<td>typchk.h</td>
<td>Defines the format of raw data in the .typchk section.</td>
</tr>
<tr>
<td>exceptab.h</td>
<td>Defines the format of raw data in the .except section.</td>
</tr>
<tr>
<td>debug.h</td>
<td>Defines the format of raw data in the .debug section.</td>
</tr>
<tr>
<td>reloc.h</td>
<td>Defines the relocation information.</td>
</tr>
<tr>
<td>linenum.h</td>
<td>Defines the line number information.</td>
</tr>
<tr>
<td>syms.h</td>
<td>Defines the symbol table format.</td>
</tr>
<tr>
<td>storclass.h</td>
<td>Defines ordinary storage classes.</td>
</tr>
<tr>
<td>dbxstclass.h</td>
<td>Defines storage classes used by the symbolic debuggers.</td>
</tr>
</tbody>
</table>

The a.out.h file includes the xcoff.h file. All of the XCOFF include files include the xcoff32_64.h file.

For more information on sections of the XCOFF object file, see "Sections and Section Headers." For more information on the symbol table, see "Symbol Table Information." For more information on the string table, see "String Table." For more information on the Debug section, see "Debug Section."

Composite File Header
The following sections describe the XCOFF composite file header components:
• File Header (filehdr.h)
• Auxiliary Header (aouthdr.h)
• Section Headers (scnhdr.h)

File Header (filehdr.h)
The filehdr.h file defines the file header of an XCOFF file. The file header is 20 bytes long in XCOFF32 and 24 bytes long in XCOFF64. The structure contains the fields shown in the following table.
Table 13. File Header Structure (Defined in filehdr.h)

<table>
<thead>
<tr>
<th>Field Name and Description</th>
<th>XCOFF32</th>
<th>XCOFF64</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>f_magic</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Target machine</td>
<td>Offset: 0</td>
<td>Offset: 0</td>
</tr>
<tr>
<td></td>
<td>Length: 2</td>
<td>Length: 2</td>
</tr>
<tr>
<td><code>f_nscns</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of sections</td>
<td>Offset: 2</td>
<td>Offset: 2</td>
</tr>
<tr>
<td></td>
<td>Length: 2</td>
<td>Length: 2</td>
</tr>
<tr>
<td><code>f_timdat</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time and date of file creation</td>
<td>Offset: 4</td>
<td>Offset: 4</td>
</tr>
<tr>
<td></td>
<td>Length: 4</td>
<td>Length: 4</td>
</tr>
<tr>
<td><code>f_symptr</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Byte offset to symbol table start</td>
<td>Offset: 8</td>
<td>Offset: 8</td>
</tr>
<tr>
<td></td>
<td>Length: 4</td>
<td>Length: 8</td>
</tr>
<tr>
<td><code>f_nsyms</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of entries in symbol table</td>
<td>Offset: 12</td>
<td>Offset: 20</td>
</tr>
<tr>
<td></td>
<td>Length: 4</td>
<td>Length: 4</td>
</tr>
<tr>
<td><code>f_opthdr</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of bytes in optional header</td>
<td>Offset: 16</td>
<td>Offset: 16</td>
</tr>
<tr>
<td></td>
<td>Length: 2</td>
<td>Length: 2</td>
</tr>
<tr>
<td><code>f_flags</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flags (see “Field Definitions”)</td>
<td>Offset: 18</td>
<td>Offset: 18</td>
</tr>
<tr>
<td></td>
<td>Length: 2</td>
<td>Length: 2</td>
</tr>
</tbody>
</table>

† Use “32” or “64” suffix when `__XCOFF_HYBRID__` is defined.

**Field Definitions:**

- `f_magic`: Specifies an integer known as the magic number, which specifies the target machine and environment of the object file. For XCOFF32, the only valid value is 0x01DF (0737 Octal). For XCOFF64 on AIX 4.3, the only valid value is 0x01EF (0757 Octal). For XCOFF64 on AIX 5.1 and later, the only valid value is 0x01F7 (0767 Octal). Symbolic names for these values are found in the file, `/usr/include/filehdr.h`.

- `f_nscns`: Specifies the number of section headers contained in the file. The first section header is section header number one; all references to a section are one-based.

- `f_timdat`: Specifies when the file was created (number of elapsed seconds since 00:00:00 Universal Coordinated Time (UCT), January 1, 1970). This field should specify either the actual time or be set to a value of 0.

- `f_symptr`: Specifies a file pointer (byte offset from the beginning of the file) to the start of the symbol table. If the value of the `f_nsyms` field is 0, then `f_nsyms` is undefined.

- `f_nsyms`: Specifies the number of entries in the symbol table. Each symbol table entry is 18 bytes long.

- `f_opthdr`: Specifies the length, in bytes, of the auxiliary header. For an XCOFF file to be executable, the auxiliary header must exist and be `_AOUTHSZ_EXEC` bytes long. (`_AOUTHSZ_EXEC` is defined in aouthdr.h.)
f_flags

Specifies a bit mask of flags that describe the type of the object file. The following information defines the flags:

<table>
<thead>
<tr>
<th>Bit Mask</th>
<th>Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0001</td>
<td>F_RELFLG</td>
<td>Indicates that the relocation information for binding has been removed from the file. This flag must not be set by compilers, even if relocation information was not required.</td>
</tr>
<tr>
<td>0x0002</td>
<td>F_EXEC</td>
<td>Indicates that the file is executable. No unresolved external references exist.</td>
</tr>
<tr>
<td>0x0004</td>
<td>F_LNNO</td>
<td>Indicates that line numbers have been stripped from the file by a utility program. This flag is not set by compilers, even if no line-number information has been generated.</td>
</tr>
<tr>
<td>0x0008</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0x0010</td>
<td>F_FDPR_PROF</td>
<td>Indicates that the file was profiled with the <code>fdpr</code> command.</td>
</tr>
<tr>
<td>0x0020</td>
<td>F_FDPR_OPTI</td>
<td>Indicates that the file was reordered with the <code>fdpr</code> command.</td>
</tr>
<tr>
<td>0x0040</td>
<td>F_DSA</td>
<td>Indicates that the file uses Very Large Program Support.</td>
</tr>
<tr>
<td>0x0080</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0x0100</td>
<td>F_VARPG</td>
<td>Indicates that one of the members of the auxiliary header specifying the medium page sizes is non-zero. By default, the value of this bit is always zero.</td>
</tr>
<tr>
<td>0x0200</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0x0400</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0x0800</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0x1000</td>
<td>F_DYNLOAD</td>
<td>Indicates the file is dynamically loadable and executable. External references are resolved by way of imports, and the file might contain exports and loader relocation.</td>
</tr>
<tr>
<td>0x2000</td>
<td>F_SHROBJ</td>
<td>Indicates the file is a shared object (shared library). The file is separately loadable. That is, it is not normally bound with other objects, and its loader exports symbols are used as automatic import symbols for other object files.</td>
</tr>
<tr>
<td>0x4000</td>
<td>F_LOADONLY</td>
<td>If the object file is a member of an archive, it can be loaded by the system loader, but the member is ignored by the binder. If the object file is not in an archive, this flag has no effect.</td>
</tr>
<tr>
<td>0x8000</td>
<td>Reserved</td>
<td></td>
</tr>
</tbody>
</table>

**Auxiliary Header (aouthdr.h)**

The auxiliary header contains system-dependent and implementation-dependent information, which is used for loading and executing a module. Information in the auxiliary header minimizes how much of the file must be processed by the system loader at execution time.
The binder generates an auxiliary header for use by the system loader. Auxiliary headers are not required for an object file that is not to be loaded. When auxiliary headers are generated by compilers and assemblers, the headers are ignored by the binder.

The auxiliary header immediately follows the file header.

**Note:** If the value of the f_opt hdr field in the file header is 0, the auxiliary header does not exist.

The C language structure for the auxiliary header is defined in the `aouthdr.h` file. The auxiliary header contains the fields shown in the following table.

**Table 14. Auxiliary Header Structure (Defined in aouthdr.h)**

<table>
<thead>
<tr>
<th>Field Name and Description</th>
<th>XCOFF32</th>
<th>XCOFF64</th>
</tr>
</thead>
<tbody>
<tr>
<td>o_mflag</td>
<td>• Offset: 0</td>
<td>• Offset: 0</td>
</tr>
<tr>
<td></td>
<td>• Length: 2</td>
<td>• Length: 2</td>
</tr>
<tr>
<td>o_vstamp</td>
<td>• Offset: 2</td>
<td>• Offset: 2</td>
</tr>
<tr>
<td></td>
<td>• Length: 2</td>
<td>• Length: 2</td>
</tr>
<tr>
<td>o_tsize</td>
<td>• Offset: 4</td>
<td>• Offset: 56</td>
</tr>
<tr>
<td>Text size in bytes</td>
<td>• Length: 4</td>
<td>• Length: 8</td>
</tr>
<tr>
<td>o_dsize</td>
<td>• Offset: 8</td>
<td>• Offset: 64</td>
</tr>
<tr>
<td>Initialized data size in bytes</td>
<td>• Length: 4</td>
<td>• Length: 8</td>
</tr>
<tr>
<td>o_bsize</td>
<td>• Offset: 12</td>
<td>• Offset: 72</td>
</tr>
<tr>
<td>Uninitialized data size in bytes</td>
<td>• Length: 4</td>
<td>• Length: 8</td>
</tr>
<tr>
<td>o_entry</td>
<td>• Offset: 16</td>
<td>• Offset: 80</td>
</tr>
<tr>
<td>Entry point descriptor (virtual address)</td>
<td>• Length: 4</td>
<td>• Length: 8</td>
</tr>
<tr>
<td>o_text_start</td>
<td>• Offset: 20</td>
<td>• Offset: 8</td>
</tr>
<tr>
<td>Base address of text (virtual address)</td>
<td>• Length: 4</td>
<td>• Length: 8</td>
</tr>
<tr>
<td>o_data_start</td>
<td>• Offset: 24</td>
<td>• Offset: 16</td>
</tr>
<tr>
<td>Base address of data (virtual address)</td>
<td>• Length: 4</td>
<td>• Length: 8</td>
</tr>
<tr>
<td>o_toc</td>
<td>• Offset: 28</td>
<td>• Offset: 24</td>
</tr>
<tr>
<td>Address of TOC anchor</td>
<td>• Length: 4</td>
<td>• Length: 8</td>
</tr>
<tr>
<td>o_sentry</td>
<td>• Offset: 32</td>
<td>• Offset: 32</td>
</tr>
<tr>
<td>Section number for entry point</td>
<td>• Length: 2</td>
<td>• Length: 2</td>
</tr>
<tr>
<td>o_sntext</td>
<td>• Offset: 34</td>
<td>• Offset: 34</td>
</tr>
<tr>
<td>Section number for .text</td>
<td>• Length: 2</td>
<td>• Length: 2</td>
</tr>
<tr>
<td>o_sndata</td>
<td>• Offset: 36</td>
<td>• Offset: 36</td>
</tr>
<tr>
<td>Section number for .data</td>
<td>• Length: 2</td>
<td>• Length: 2</td>
</tr>
<tr>
<td>o_sntoc</td>
<td>• Offset: 38</td>
<td>• Offset: 38</td>
</tr>
<tr>
<td>Section number for TOC</td>
<td>• Length: 2</td>
<td>• Length: 2</td>
</tr>
</tbody>
</table>
Table 14. Auxiliary Header Structure (Defined in aouthdr.h) (continued)

<table>
<thead>
<tr>
<th>Field Name and Description</th>
<th>XCOFF32</th>
<th>XCOFF64</th>
</tr>
</thead>
</table>
| **o_snloader**             | • Offset: 40  
                             | • Length: 2   | • Offset: 40  
                             | • Length: 2   |
| Section number for loader data |         |         |
| **o_snbs**                 | • Offset: 42  
                             | • Length: 2   | • Offset: 42  
                             | • Length: 2   |
| Section number for .bss   |         |         |
| **o_alignntext**           | • Offset: 44  
                             | • Length: 2   | • Offset: 44  
                             | • Length: 2   |
| Maximum alignment for .text |         |         |
| **o_alignndata**           | • Offset: 46  
                             | • Length: 2   | • Offset: 46  
                             | • Length: 2   |
| Maximum alignment for .data |         |         |
| **o_modtype**              | • Offset: 48  
                             | • Length: 2   | • Offset: 48  
                             | • Length: 2   |
| Module type field         |         |         |
| **o_cpuflag**              | • Offset: 50  
                             | • Length: 1   | • Offset: 50  
                             | • Length: 1   |
| Bit flags - cpu types of objects |         |         |
| **o_cputype**              | • Offset: 51  
                             | • Length: 1   | • Offset: 51  
                             | • Length: 1   |
| Reserved for CPU type     |         |         |
| **o_maxstack**             | • Offset: 52  
                             | • Length: 4   | • Offset: 88  
                             | • Length: 8   |
| Maximum stack size allowed (bytes) |         |         |
| **o_maxdata**              | • Offset: 56  
                             | • Length: 4   | • Offset: 96  
                             | • Length: 8   |
| Maximum data size allowed (bytes) |         |         |
| **o_debugger**             | • Offset: 60  
                             | • Length: 4   | • Offset: 4    
                             | • Length: 4   |
| Reserved for debuggers.   |         |         |
| **o_flags**                | • Offset: 67  
                             | • Length: 1   | • Offset: 55  
                             | • Length: 1   |
| Flags and thread-local storage alignment |         |         |
| **o_sndata**               | • Offset: 68  
                             | • Length: 2   | • Offset: 104  
                             | • Length: 2   |
| Section number for .tdata |         |         |
| **o_sntbss**               | • Offset: 70  
                             | • Length: 2   | • Offset: 106  
                             | • Length: 2   |
| Section number for .tbss  |         |         |

+Use "32" or "64" suffix when __XCOFF_HYBRID__ is defined.

**Field Definitions:** The following information defines the auxiliary header fields. For entries with two labels, the label in parentheses is the alternate original COFF a.out file format name.

**o_mflags** (magic)  
Specifies the magic number, which informs the operating system of the file’s execution characteristics. The binder assigns the following value:

0x010B  
Text and data are aligned in the file and may be paged.

**o_vstamp** (vstamp)  
Specifies the format version for this auxiliary header. The only valid value is 1.
o_tsize (tsize) Specifies the size (in bytes) of the raw data for the .text section. The .text section typically contains the read-only part of the program. This is the same value as contained in the s_size field of the section header for the .text section.

o_dsize (dsize) Specifies the size (in bytes) of the raw data for the .data section. The .data section contains the initialized data of the program and is writable. This is the same value as contained in the s_size field of the section header for the .data section.

o_bsize (bsize) Specifies the size (in bytes) of .bss area, which is used for uninitialized variables during execution and is writable. No raw data exists in the file for the .bss section. This is the same value as contained in the s_size field of the section header for the .bss section.

o_entry (entry) Specifies the virtual address of the entry point. (See the definition of the o_snentry field.) For application programs, this virtual address is the address of the function descriptor. The function descriptor contains the addresses of both the entry point itself and its TOC anchor. The offset of the entry point function descriptor from the beginning of its containing section can be calculated as follows:

\[
\text{Section Offset Value} = o_{entry} - s_{paddr}[o_{snentry} - 1],
\]

where \( s_{paddr} \) is the virtual address contained in the specified section header.

o_text_start (text_start) Specifies the virtual address of the .text section. This is the address assigned to (that is, used for) the first byte of the .text raw-data section. This is the same value as contained in the s_paddr field of the section header for the .text section.

o_data_start (data_start) Specifies the virtual address of the .data section. This is the address assigned to (that is, used for) the first byte of the .data raw-data section. This is the same value as contained in the s_paddr field of the section header for the .data section.

For addressing purposes, the .bss section is considered to follow the .data section.

o_toc Specifies the virtual address of the TOC anchor (see the definition of the o_sntoc field).

o_snentry Specifies the number of the file section containing the entry-point. (This field contains a file section header sequence number.) The entry point must be in the .text or .data section.

o_sntext Specifies the number of the file .text section. (This field contains a file section header sequence number.)

o_sndata Specifies the number of the file .data section. (This field contains a file section header sequence number.)

o_sntoc Specifies the number of the file section containing the TOC. (This field contains a file section header sequence number.)

o_snloader Specifies the number of the file section containing the system loader information. (This field contains a file section header sequence number.)

o_snbss Specifies the number of the file .bss section. (This field contains a file section header sequence number.)

o_algntext Specifies the log (base 2) of the maximum alignment needed for any csect in the .text section.

o_algnndata Specifies the log (base 2) of the maximum alignment needed for any csect in the .data and .bss sections.

o_modtype Specifies a module type. The value is an ASCII character string. The following module type is recognized by the system loader:

RO Specifies a read-only module. If a shared object with this module type has no BSS section and only depends on other read-only modules, the data section of the module will be mapped read-only and shared by all processes using the object.

o_cpuflag Bit flags - cputypes of objects.

o_cptype Reserved. This byte must be set to 0.

o_maxstack Specifies the maximum stack size (in bytes) allowed for this executable. If the value is 0, the system default maximum stack size is used.

o_maxdata Specifies the maximum data size (in bytes) allowed for this executable. If the value is 0, the system default maximum data size is used.

o_debugger This field should contain 0. When a loaded program is being debugged, the memory image of this field may be modified by a debugger to insert a trap instruction.

o_sndata Specifies the number of the .tdata file section. (This field contains a file section header sequence number.)
**o_sntbss** Specifies the number of the .tbss file section. (This field contains a file section header sequence number.)

**o_flags** Consists of four 1-bit flags and a 4-bit .tdata alignment:

- **_AOUT_TLS_LE 0x80** (High-order bit of o_flags)
  Program uses local-exec model for thread-local storage. Such a program cannot be loaded dynamically.

- **_AOUT_ALGNTDATA (Low-order 4 bits of o_flags)**
  This field specifies the desired alignment of the module’s thread-local storage. The value of this 4-bit number is interpreted as follows:
  - Bit 0-8: Log (base 2) of desired alignment
  - Bit 9-11: Reserved.
  - Bit 12: 4KB page alignment
  - Bit 13: 64KB page alignment
  - Bit 14-15: Reserved.

**o_textpsize** Specifies the size of pages for the exec text. The default value is 0 (system-selected page size).

**o_datapsize** Specifies the size of pages for the exec data. The default value is 0 (system-selected page size). The value of o_datapsize overrides the large page data request (by setting the F_LPDATA bit in the XCOFF file).

**o_stackpsize** Specifies the size of pages for the stack. The default value is 0 (system-selected page size).

In general, an object file might contain multiple sections of a given type, but in a loadable module, there must be exactly one .text, .data, .bss, and .loader section. A loadable object might also have one .tdata section and one .tbss section.

### Section Headers (scnhdr.h)

Each section of an XCOFF file has a corresponding section header, although some section headers may not have a corresponding raw-data section. A section header provides identification and file-accessing information for each section contained within an XCOFF file. Each section header in an XCOFF32 file is 40 bytes long, while XCOFF64 section headers are 72 bytes long. The C language structure for a section header can be found in the scnhdr.h file. A section header contains the fields shown in the following table.

**Table 15. Section Header Structure (Defined in scnhdr.h)**

<table>
<thead>
<tr>
<th>Field Name and Description</th>
<th>XCOFF32</th>
<th>XCOFF64</th>
</tr>
</thead>
</table>
| s_name                    | • Offset: 0  
                           | • Length: 8  
                           | • Offset: 0  
                           | • Length: 8  |
| s_paddr                   | • Offset: 8  
                           | • Length: 4  
                           | • Offset: 8  
                           | • Length: 8  |
| s_vaddr                   | • Offset: 12 
                           | • Length: 4  
                           | • Offset: 16 
                           | • Length: 8  |
| s_size                    | • Offset: 16 
                           | • Length: 4  
                           | • Offset: 24 
                           | • Length: 8  |
| s_scnptr                  | • Offset: 20 
                           | • Length: 4  
                           | • Offset: 32 
                           | • Length: 8  |
| s_relptr                  | • Offset: 24 
                           | • Length: 4  
                           | • Offset: 40 
                           | • Length: 8  |
### Field Definitions

The following information defines the section header fields:

- **s_name**: Specifies an 8-byte, null-padded section name. An 8-byte section name will not have a terminating null character. Use the `s_flags` field instead of the `s_name` field to determine a section type. Two sections of the same type may have different names, allowing certain applications to distinguish between them.

- **s_paddr**: Specifies the physical address of the section. This is the address assigned and used by the compilers and the binder for the first byte of the section. This field should contain 0 for all sections except the `.text`, `.data`, and `.bss` sections.

- **s_vaddr**: Specifies the virtual address of the section. This field has the same value as the `s_paddr` field.

- **s_size**: Specifies the size (in bytes) of this section.

- **s_scnptr**: Specifies a file pointer (byte offset from the beginning of the file) to this section’s raw data. If this field contains 0, this section has no raw data. Otherwise, the size of the raw data must be contained in the `s_size` field.

- **s_relptr**: Specifies a file pointer (byte offset from the beginning of the file) to the relocation entries for this section. If this section has no relocation entries, this field must contain 0.

- **s_lnnoptr**: Specifies a file pointer (byte offset from the beginning of the file) to the line number entries for this section. If this section has no line number entries, this field must contain 0.

- **s_nreloc**: Specifies the number of relocation entries for this section. In an XCOFF32 file, if more than 65,534 relocation entries are required, the field value will be 65535, and an `STYP_OVRFLO` section header will contain the actual count of relocation entries in the `s_paddr` field. Refer to the discussion of overflow headers in "Sections and Section Headers". If this field is set to 65535, the `s_nlnno` field must also be set to 65535.

- **s_nlnno**: Specifies the number of line number entries for this section. In an XCOFF32 file, if more than 65,534 line number entries are required, the field value will be 65535, and an `STYP_OVRFLO` section header will contain the actual number of line number entries in the `s_vaddr` field. Refer to the discussion of overflow headers in "Sections and Section Headers". If this field is set to 65535, the `s_nreloc` field must also be set to 65535.

---

<table>
<thead>
<tr>
<th>Field Name and Description</th>
<th>XCOFF32</th>
<th>XCOFF64</th>
</tr>
</thead>
<tbody>
<tr>
<td>s_lnnoptr*</td>
<td>Offset: 28</td>
<td>Offset: 48</td>
</tr>
<tr>
<td>Offset in file to line number entries for section</td>
<td>Length: 4</td>
<td>Length: 8</td>
</tr>
<tr>
<td>s_nrelc*</td>
<td>Offset: 32</td>
<td>Offset: 56</td>
</tr>
<tr>
<td>Number of relocation entries</td>
<td>Length: 2</td>
<td>Length: 4</td>
</tr>
<tr>
<td>s_nlnno*</td>
<td>Offset: 34</td>
<td>Offset: 60</td>
</tr>
<tr>
<td>Number of line number entries</td>
<td>Length: 2</td>
<td>Length: 4</td>
</tr>
<tr>
<td>s_flags*</td>
<td>Offset: 36</td>
<td>Offset: 64</td>
</tr>
<tr>
<td>Flags to define the section type</td>
<td>Length: 2</td>
<td>Length: 4</td>
</tr>
</tbody>
</table>

*Use "32" or "64" suffix when `__XCOFF_HYBRID__` is defined.
Specifies flags defining the section type. The low-order pair of bytes is used. A section type identifies the contents of a section and specifies how the section is to be processed by the binder or the system loader. Only a single bit value may be assigned to the $s_{\text{flags}}$ field. This value must not be the sum or bitwise OR of multiple flags. The two high-order bytes should contain 0.

Valid bit values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Flag</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>Reserved.</td>
</tr>
<tr>
<td>0x0001</td>
<td>Reserved.</td>
</tr>
<tr>
<td>0x0002</td>
<td>Reserved.</td>
</tr>
<tr>
<td>0x0004</td>
<td>Reserved.</td>
</tr>
<tr>
<td>0x0008</td>
<td>STYP_PAD</td>
</tr>
</tbody>
</table>

Specifies a pad section. A section of this type is used to provide alignment padding between sections within an XCOFF executable object file. This section header type is obsolete since padding is allowed in an XCOFF file without a corresponding pad section header.

<table>
<thead>
<tr>
<th>Value</th>
<th>Flag</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0010</td>
<td>Reserved.</td>
</tr>
<tr>
<td>0x0020</td>
<td>STYP_TEXT</td>
</tr>
</tbody>
</table>

Specifies an executable text (code) section. A section of this type contains the executable instructions of a program.

<table>
<thead>
<tr>
<th>Value</th>
<th>Flag</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0040</td>
<td>STYP_DATA</td>
</tr>
</tbody>
</table>

Specifies an initialized data section. A section of this type contains the initialized data and the TOC of a program.

<table>
<thead>
<tr>
<th>Value</th>
<th>Flag</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0080</td>
<td>STYP_BSS</td>
</tr>
</tbody>
</table>

Specifies an uninitialized data section. A section header of this type defines the uninitialized data of a program.

<table>
<thead>
<tr>
<th>Value</th>
<th>Flag</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0100</td>
<td>STYP_EXCEPT</td>
</tr>
</tbody>
</table>

Specifies an exception section. A section of this type provides information to identify the reason that a trap or exception occurred within an executable object program.

<table>
<thead>
<tr>
<th>Value</th>
<th>Flag</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0200</td>
<td>STYP_INFO</td>
</tr>
</tbody>
</table>

Specifies a comment section. A section of this type provides comments or data to special processing utility programs.

<table>
<thead>
<tr>
<th>Value</th>
<th>Flag</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0400</td>
<td>STYP_TDATA</td>
</tr>
</tbody>
</table>

Specifies an initialized thread-local data section.

<table>
<thead>
<tr>
<th>Value</th>
<th>Flag</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0800</td>
<td>STYP_TBSS</td>
</tr>
</tbody>
</table>

Specifies an uninitialized thread-local data section.
s_flags continued

Valid bit values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Flag</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x1000</td>
<td>STYP_LOADER</td>
</tr>
<tr>
<td></td>
<td>Specifies a loader section. A section of this type contains object file information for the system loader to load an XCOFF executable. The information includes imported symbols, exported symbols, relocation data, type-check information, and shared object names.</td>
</tr>
<tr>
<td>0x2000</td>
<td>STYP_DEBUG</td>
</tr>
<tr>
<td></td>
<td>Specifies a debug section. A section of this type contains stabstring information used by the symbolic debugger.</td>
</tr>
<tr>
<td>0x4000</td>
<td>STYP_TYPCHK</td>
</tr>
<tr>
<td></td>
<td>Specifies a type-check section. A section of this type contains parameter/argument type-check strings used by the binder.</td>
</tr>
<tr>
<td>0x8000</td>
<td>STYP_OVRFLO</td>
</tr>
<tr>
<td></td>
<td>Note: An XCOFF64 file may not contain an overflow section header.</td>
</tr>
<tr>
<td></td>
<td>Specifies a relocation or line-number field overflow section. A section header of this type contains the count of relocation entries and line number entries for some other section. This section header is required when either of the counts exceeds 65,534. See the s_nreloc and s_nlnno fields in &quot;Sections and Section Headers&quot; for more information on overflow headers.</td>
</tr>
</tbody>
</table>

For general information on the XCOFF file format, see "XCOFF Object File Format."

Sections and Section Headers

Section headers are defined to provide a variety of information about the contents of an XCOFF file. Programs that process XCOFF files will recognize only some of the valid sections.

See the following information to learn more about XCOFF file sections:

- **Loader Section (loader.h)**
- **Debug Section**
- **Type-Check Section**
- **Exception Section**
- **Comment Section**

Current applications do not use the s_name field to determine the section type. Nevertheless, conventional names are used by system tools, as shown in the following table.

<table>
<thead>
<tr>
<th>Description</th>
<th>Multiple Allowed?</th>
<th>s_flag (and its conventional name)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text section</td>
<td>Yes</td>
<td>STYP_TEXT (.text)</td>
</tr>
<tr>
<td>Data section</td>
<td>Yes</td>
<td>STYP_DATA (.data)</td>
</tr>
<tr>
<td>BSS section</td>
<td>Yes</td>
<td>STYP_BSS (.bss)</td>
</tr>
<tr>
<td>Pad section</td>
<td>Yes</td>
<td>STYP_PAD (.pad)</td>
</tr>
<tr>
<td>Loader section</td>
<td>No</td>
<td>STYP_LOADER (.loader)</td>
</tr>
<tr>
<td>Debug section</td>
<td>No</td>
<td>STYP_DEBUG (.debug)</td>
</tr>
<tr>
<td>Type-check section</td>
<td>Yes</td>
<td>STYP_TYPCHK (.typchk)</td>
</tr>
<tr>
<td>Exception section</td>
<td>No</td>
<td>STYP_EXCEPT (.except)</td>
</tr>
<tr>
<td>Overflow section</td>
<td>Yes (one per .text or .data section)</td>
<td>STYP_OVRFLO (.ovrflo)</td>
</tr>
</tbody>
</table>
### Table 16. Conventional Header Names (continued)

<table>
<thead>
<tr>
<th>Description</th>
<th>Multiple Allowed?</th>
<th>s_flag (and its conventional name)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comment section</td>
<td>Yes</td>
<td>STYP_INFO (.info)</td>
</tr>
<tr>
<td>Tdata section</td>
<td>Yes</td>
<td>STYP_TDATA (.tdata)</td>
</tr>
<tr>
<td>TBSS section</td>
<td>Yes</td>
<td>STYP_TBSS (.tbss)</td>
</tr>
</tbody>
</table>

Some fields of a section header may not always be used, or may have special usage. This pertains to the following fields:

- **s_name**: On input, ignored by the binder and system loader. On output, the conventional names (shown in the "Conventional Header Names" table) are used.
- **s_scnptr**: Ignored for .bss sections.
- **s_relpotr**: Recognized for the .text, .data, and .tdata sections only. No relocation is performed for other sections, where this value must be 0.
- **s_lnnoptr**: Recognized for the .text section only. Otherwise, it must be 0.
- **s_nreloc, s_nlnno**: Handles relocation or line-number field overflows in an XCOFF32 file. (XCOFF64 files may not have overflow section headers.) If a section has more than 65,534 relocation entries or line number entries, both of these fields are set to a value of 65535. In this case, an overflow section header with the s_flags field equal to STYP_OVRFO is used to contain the relocation and line-number count information. The fields in the overflow section header are defined as follows:
  - **s_nreloc**: Specifies the file section number of the section header that overflowed; that is, the section header containing a value of 65535 in its s_nreloc and s_nlnno fields. This value provides a reference to the primary section header. This field must have the same value as the s_nlnno field.
    **Note**: There is no reference in the primary section header that identifies the appropriate overflow section header. All the section headers must be searched to locate an overflow section header that contains the correct primary section header reference in this field.
  - **s_nlnno**: Specifies the file section number of the section header that overflowed. This field must have the same value as the s_nreloc field.
  - **s_paddr**: Specifies the number of relocation entries actually required. This field is used instead of the s_nreloc field of the section header that overflowed.
  - **s_vaddr**: Specifies the number of line-number entries actually required. This field is used instead of the s_nlnno field of the section header that overflowed.

The s_size and s_scnptr fields have a value of 0 in an overflow section header. The s_relpotr and s_lnnoptr fields must have the same values as in the corresponding primary section header.

An XCOFF file provides special meaning to the following sections:

- The .text, .data, and .bss sections, and the optional .tdata and .tbss sections, define the memory image of the program. The relocation parts associated with the .text and .data sections contain the full binder relocation information so it can be used for replacement link editing. Only the .text section is associated with a line number part. The parts associated with the executable code are produced by the compilers and assemblers.
The .pad section is defined as a null-filled, raw-data section that is used to align a subsequent section in the file on some defined boundary such as a file block boundary or a system page boundary. Padding is allowed in an XCOFF file without a corresponding section header so that the binder does not generate pad section headers.

The .loader section is a raw-data section defined to contain the dynamic loader information. This section is generated by the binder and has its own self-contained symbol table and relocation table. There is no reference to this section from the XCOFF Symbol Table.

The .debug section is a raw-data section defined to contain the stab (symbol table) or dictionary information required by the symbolic debugger.

The .typchk section is a raw-data section defined to contain parameter and argument type-checking strings.

The .except section is a raw-data section defined to contain the exception tables used to identify the reasons for an exception in program execution.

The .info comment section is a raw-data section defined to contain comments or data that are of significance to special processing utility programs.

The .debug, .except, .info, and .typchk sections are produced by compilers and assemblers. References to these sections or to items within these sections are made from the XCOFF Symbol Table.

For more information on XCOFF file sections, see "Loader Section (loader.h)," "Debug Section," "Type-Check Section," "Exception Section," and "Comment Section."

### Loader Section (loader.h)

The loader section contains information required by the system loader to load and relocate an executable XCOFF object. The loader section is generated by the binder. The loader section has an s_flags section type flag of STYP_LOADER in the XCOFF section header. By convention, .loader is the loader section name. The data in this section is not referenced by entries in the XCOFF symbol table.

The loader section consists of the following parts:

- Header fields
- Symbol table
- Relocation table
- Import file ID strings
- Symbol name string table

The C language structure for the loader section can be found in the loader.h file.

#### Loader Header Field Definitions

The following table describes the loader section’s header field definitions.

<table>
<thead>
<tr>
<th>Field Name and Description</th>
<th>XCOFF32</th>
<th>XCOFF64</th>
</tr>
</thead>
<tbody>
<tr>
<td>l_version</td>
<td>Offset: 0</td>
<td>Offset: 0</td>
</tr>
<tr>
<td></td>
<td>Length: 4</td>
<td>Length: 4</td>
</tr>
<tr>
<td>l_nsyms</td>
<td>Offset: 4</td>
<td>Offset: 4</td>
</tr>
<tr>
<td></td>
<td>Length: 4</td>
<td>Length: 4</td>
</tr>
</tbody>
</table>
Table 17. Loader Section Header Structure (Defined in loader.h) (continued)

<table>
<thead>
<tr>
<th>Field Name and Description</th>
<th>XCOFF32</th>
<th>XCOFF64</th>
</tr>
</thead>
<tbody>
<tr>
<td>l_nreloc</td>
<td>-</td>
<td>Number of relocation table entries</td>
</tr>
<tr>
<td></td>
<td>• Offset: 8</td>
<td>• Offset: 8</td>
</tr>
<tr>
<td></td>
<td>• Length: 4</td>
<td>• Length: 4</td>
</tr>
<tr>
<td>l_istlen</td>
<td>-</td>
<td>Length of import file ID string table</td>
</tr>
<tr>
<td></td>
<td>• Offset: 12</td>
<td>• Offset: 12</td>
</tr>
<tr>
<td></td>
<td>• Length: 4</td>
<td>• Length: 4</td>
</tr>
<tr>
<td>l_nimpid</td>
<td>-</td>
<td>Number of import file IDs</td>
</tr>
<tr>
<td></td>
<td>• Offset: 16</td>
<td>• Offset: 16</td>
</tr>
<tr>
<td></td>
<td>• Length: 4</td>
<td>• Length: 4</td>
</tr>
<tr>
<td>l_impoff</td>
<td>-</td>
<td>Offset to start of import file IDs</td>
</tr>
<tr>
<td></td>
<td>• Offset: 20</td>
<td>• Offset: 24</td>
</tr>
<tr>
<td></td>
<td>• Length: 4</td>
<td>• Length: 8</td>
</tr>
<tr>
<td>l_stlen</td>
<td>-</td>
<td>Length of string table</td>
</tr>
<tr>
<td></td>
<td>• Offset: 24</td>
<td>• Offset: 20</td>
</tr>
<tr>
<td></td>
<td>• Length: 4</td>
<td>• Length: 4</td>
</tr>
<tr>
<td>l_stoff</td>
<td>-</td>
<td>Offset to start of string table</td>
</tr>
<tr>
<td></td>
<td>• Offset: 28</td>
<td>• Offset: 32</td>
</tr>
<tr>
<td></td>
<td>• Length: 4</td>
<td>• Length: 8</td>
</tr>
<tr>
<td>l_symoff</td>
<td>-</td>
<td>Offset to start of symbol table</td>
</tr>
<tr>
<td></td>
<td>• Offset: N/A</td>
<td>• Offset: 40</td>
</tr>
<tr>
<td></td>
<td>• Length: N/A</td>
<td>• Length: 8</td>
</tr>
<tr>
<td>l_rldoff</td>
<td>-</td>
<td>Offset to start of relocation entries</td>
</tr>
<tr>
<td></td>
<td>• Offset: 36</td>
<td>• Offset: 64</td>
</tr>
<tr>
<td></td>
<td>• Length: 2</td>
<td>• Length: 4</td>
</tr>
</tbody>
</table>

*Use “32” or “64” suffix when __XCOFF_HYBRID__ is defined.

The following information defines the loader section’s header fields:

- l_version: Specifies the loader section version number. This value must be 1 for XCOFF32, 2 for XCOFF64.
- l_nsym: Specifies the number of symbol table entries in the loader section. This value is the actual count of symbol table entries contained in the loader section and does not include the three implicit entries for the .text, .data, and .bss symbol entries.
- l_nreloc: Specifies the number of relocation table entries in the loader section.
- l_istlen: Specifies the byte length of the import file ID string table in the loader section.
- l_nimpid: Specifies the number of import file IDs in the import file ID string table.
- l_impoff: Specifies the byte offset from beginning of the loader section to the first import file ID.
- l_stlen: Specifies the length of the loader section string table.
- l_stoff: Specifies the byte offset from beginning of the loader section to the first entry in the string table.
- l_symoff: Specifies the byte offset from beginning of the loader section to the start of the loader symbol table (in XCOFF64 only).
- l_rldoff: Specifies the byte offset from beginning of the loader section to the start of the loader section relocation entries (in XCOFF64 only).

Loader Symbol Table Field Definitions

The loader section symbol table contains the symbol table entries that the system loader needs for its import and export symbol processing and dynamic relocation processing.

The loader.h file defines the symbol table fields. Each entry is 24 bytes long.
There are five implicit external symbols, one each for the .text, .data, .bss, .tdata, and .tbss sections. These symbols are referenced from the relocation table entries using symbol table index values 0, 1, 2, -1, and -2, respectively.

Table 18. Loader Section Symbol Table Entry Structure

<table>
<thead>
<tr>
<th>Field Name and Description</th>
<th>XCOFF32</th>
<th>XCOFF64</th>
</tr>
</thead>
<tbody>
<tr>
<td>l_name</td>
<td>Offset: 0</td>
<td>Offset: N/A</td>
</tr>
<tr>
<td></td>
<td>Length: 8</td>
<td>Length: N/A</td>
</tr>
<tr>
<td>l_zeroes</td>
<td>Offset: 0</td>
<td>Offset: N/A</td>
</tr>
<tr>
<td></td>
<td>Length: 4</td>
<td>Length: N/A</td>
</tr>
<tr>
<td>l_offset</td>
<td>Offset: 4</td>
<td>Offset: 8</td>
</tr>
<tr>
<td></td>
<td>Length: 4</td>
<td>Length: 4</td>
</tr>
<tr>
<td>l_value</td>
<td>Offset: 8</td>
<td>Offset: 0</td>
</tr>
<tr>
<td></td>
<td>Length: 4</td>
<td>Length: 8</td>
</tr>
<tr>
<td>l_scnum</td>
<td>Offset: 12</td>
<td>Offset: 12</td>
</tr>
<tr>
<td></td>
<td>Length: 2</td>
<td>Length: 2</td>
</tr>
<tr>
<td>l_smtyp</td>
<td>Offset: 14</td>
<td>Offset: 14</td>
</tr>
<tr>
<td></td>
<td>Length: 1</td>
<td>Length: 1</td>
</tr>
<tr>
<td>l_smclas</td>
<td>Offset: 15</td>
<td>Offset: 15</td>
</tr>
<tr>
<td></td>
<td>Length: 1</td>
<td>Length: 1</td>
</tr>
<tr>
<td>l_ifile</td>
<td>Offset: 16</td>
<td>Offset: 16</td>
</tr>
<tr>
<td></td>
<td>Length: 4</td>
<td>Length: 4</td>
</tr>
<tr>
<td>l_parm</td>
<td>Offset:20</td>
<td>Offset: 20</td>
</tr>
<tr>
<td></td>
<td>Length: 4</td>
<td>Length: 4</td>
</tr>
</tbody>
</table>

*Use “32” or “64” suffix when __XCOFF_HYBRID__ is defined.*

The symbol table fields are:

- **l_name** (XCOFF32 only) Specifies an 8-byte, null-padded symbol name if it is 8 bytes or less in length. Otherwise, the field is treated as the following two 4-byte integers for accessing the symbol name:
  - **l_zeroes** (XCOFF32 only) A value of 0 indicates that the symbol name is in the loader section string table. This field overlays the first word of the l_name field. An l_name field having the first 4 bytes (first word) equal to 0 is used to indicate that the name string is contained in the string table instead of the l_name field.
  - **l_offset** (XCOFF32 only) This field overlays the second word of the l_name field. The value of this field is the byte offset from the beginning of the loader section string table to the first byte of the symbol name (not its length field).

- **l_offset** (XCOFF64 only) This field has the same use as the l_offset field in XCOFF32.
- **l_value** Specifies the virtual address of the symbol.
Specifies the number of the XCOFF section that contains the symbol. If the symbol is undefined or imported, the section number is 0. Otherwise, the section number refers to the .text, .data, or .bss section. Section headers are numbered beginning with 1.

Specifies the symbol type, import flag, export flag, and entry flag.

Bits 0-4 are flag bits defined as follows:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0x80</td>
<td>Reserved</td>
</tr>
<tr>
<td>1</td>
<td>0x40</td>
<td>Specifies an imported symbol.</td>
</tr>
<tr>
<td>2</td>
<td>0x20</td>
<td>Specifies an entry point descriptor symbol.</td>
</tr>
<tr>
<td>3</td>
<td>0x10</td>
<td>Specifies an exported symbol.</td>
</tr>
<tr>
<td>4</td>
<td>0x08</td>
<td>Specifies a weak symbol.</td>
</tr>
</tbody>
</table>

Bits 5-7 constitute a 3-bit symbol type field with the following definitions:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Symbol Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>XTY_ER</td>
</tr>
<tr>
<td>1</td>
<td>XTY_SD</td>
</tr>
<tr>
<td>2</td>
<td>XTY_LD</td>
</tr>
<tr>
<td>3</td>
<td>XTY_CM</td>
</tr>
<tr>
<td>4-7</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Specifies an external reference providing a symbol table entry for an external (global) symbol contained in another XCOFF object file.

Specifies the csect section definition, providing the definition of the smallest initialized unit within an XCOFF object file.

Specifies the label definition, providing the definition of the global entry points for initialized csects. An uninitialized csect of type XTY_CM may not contain a label definition.

Specifies a common (BSS uninitialized data) csect definition, providing the definition of the smallest uninitialized unit within an XCOFF object file.

Reserved.

Specifies the storage mapping class of the symbol, as defined in syms.h for the x_smclas field of the csect auxiliary symbol table entry. Values have the symbolic form XMC_xx, where xx is PR, RO, GL, XO, SV, SV64, SV3264, RW, TC, TD, DS, UA, BS, UC, TL, or UL. See [csect Auxiliary Entry for the C_EXT, C_WEAKEXT, and C_HIDEXT Symbols] for more information.

Specifies the import file ID string. This integer is the ordinal value of the position of the import file ID string in the import file ID name string table of the loader section. For an imported symbol, the value of 0 in this field identifies the symbol as a deferred import to the system loader. A deferred import is a symbol whose address can remain unresolved following the processing of the loader. If the symbol was not imported, this field must have a value of 0.

Specifies the offset to the parameter type-check string. The byte offset is from the beginning of the loader section string table. The byte offset points to the first byte of the parameter type-check string (not to its length field). For more information on the parameter type-check string, see [Type-Check Section]. A value of 0 in the l_parm field indicates that the parameter type-checking string is not present for this symbol, and the symbol will be treated as having a universal hash.

Loader Relocation Table Field Definitions

The Loader Section Relocation Table Structure contains all the relocation information that the system loader needs to properly relocate an executable XCOFF file when it is loaded. The loader.h file defines the relocation table fields. Each entry in the loader section relocation table is 12 bytes long in XCOFF32 and 16 bytes long in XCOFF64. The l_vaddr, l_symndx, and l_rtype fields have the same meaning as the corresponding fields of the regular relocation entries, which are defined in the reloc.h file. See [Relocation Information for XCOFF File (reloc.h)] for more information.
Table 19. Loader Section Relocation Table Entry Structure

<table>
<thead>
<tr>
<th>Field Name and Description</th>
<th>XCOFF32</th>
<th>XCOFF64</th>
</tr>
</thead>
<tbody>
<tr>
<td>l_vaddr*</td>
<td>Offset: 0</td>
<td>Offset: 0</td>
</tr>
<tr>
<td>Address field</td>
<td>Length: 4</td>
<td>Length: 8</td>
</tr>
<tr>
<td>l_symndx*</td>
<td>Offset: 4</td>
<td>Offset: 12</td>
</tr>
<tr>
<td>Loader section symbol table index of referenced item</td>
<td>Length: 4</td>
<td>Length: 4</td>
</tr>
<tr>
<td>l_rtype</td>
<td>Offset: 4</td>
<td>Offset: 8</td>
</tr>
<tr>
<td>Relocation type</td>
<td>Length: 4</td>
<td>Length: 4</td>
</tr>
<tr>
<td>l_value*</td>
<td>Offset: 8</td>
<td>Offset: 8</td>
</tr>
<tr>
<td>Address field</td>
<td>Length: 2</td>
<td>Length: 2</td>
</tr>
<tr>
<td>l_rsecnm</td>
<td>Offset: 10</td>
<td>Offset: 10</td>
</tr>
<tr>
<td>File section number being relocated</td>
<td>Length: 2</td>
<td>Length: 2</td>
</tr>
</tbody>
</table>

+Use “32” or “64” suffix when __XCOFF_HYBRID__ is defined.

The loader.h file defines the following fields:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>l_vaddr</td>
<td>Specifies the virtual address of the relocatable reference.</td>
</tr>
<tr>
<td>l_symndx</td>
<td>Specifies the loader section symbol table index (n-th entry) of the symbol that is being referenced. Values 0, 1, and 2 are implicit references to the .text, .data, and .bss sections, respectively. Symbol index 3 is the index for the first symbol actually contained in the loader section symbol table. <strong>Note:</strong> A reference to an exported symbol can be made using the symbol’s section number (symbol number 0, 1, or 2) or using the actual number of the exported symbol.</td>
</tr>
<tr>
<td>l_rtype</td>
<td>Specifies the relocation size and type. (This field has the same interpretation as the r_type field in the reloc.h file.) See Relocation Information for XCOFF File (reloc.h) for more information.</td>
</tr>
<tr>
<td>l_rsecnm</td>
<td>Specifies the section number of the .text, .data, or .bss section being relocated (associated with l_vaddr field). This is a one-based index into the section headers.</td>
</tr>
</tbody>
</table>

Loader Import File ID Name Table Definition

The loader section import file ID name strings of a module provide a list of dependent modules that the system loader must load in order for the module to load successfully. However, this list does not contain the names of modules that the named modules themselves depend on.

Table 20. Loader Section Import File IDs - Contains Variable Length Strings

<table>
<thead>
<tr>
<th>Offset</th>
<th>Length in Bytes</th>
<th>Name and Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>n1</td>
<td>l_impidpath</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Import file ID path string, null-delimited</td>
</tr>
<tr>
<td>n1 + 1</td>
<td>n2</td>
<td>l_impidbase</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Import file ID base string, null-delimited</td>
</tr>
<tr>
<td>n1 + n2 + 2</td>
<td>n3</td>
<td>l_impidmem</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Import file ID member string, null-delimited</td>
</tr>
</tbody>
</table>

Fields repeat for each import file ID.

Each import file ID name consists of three null-delimited strings.
The first import file ID is a default LIBPATH value to be used by the system loader. The LIBPATH information consists of file paths separated by colons. There is no base name or archive member name, so the file path is followed by three null bytes.

Each entry in the import file ID name table consists of:
- Import file ID path name
- Null delimiter (ASCII Null Character)
- Import file ID base name
- Null delimiter
- Import file ID archive-file-member name
- Null delimiter

For example:
/usr/lib\0mylib.a\0shr.o\0

**Loader String Table Definition**

The loader section string table contains the parameter type-checking strings, all symbols names for an XCOFF64 file, and the names of symbols longer than 8 bytes for an XCOFF32 file. Each string consists of a 2-byte length field followed by the string.

**Table 21. Loader Section String Table**

<table>
<thead>
<tr>
<th>Offset</th>
<th>Length in Bytes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>Length of string.</td>
</tr>
<tr>
<td>2</td>
<td>n</td>
<td>Symbol name string (null-delimited) or parameter type string (not null-delimited).</td>
</tr>
</tbody>
</table>

Symbol names are null-terminated. The value in the length-field includes the length of the string plus the length of the null terminator but does not include the length of the length field itself.

The parameter type-checking strings contain binary values and are not null-terminated. The value in the length field includes the length of the string only but does not include the length of the length field itself.

The symbol table entries of the loader section contain a byte offset value that points to the first byte of the string instead of to the length field.

**Loader Section Header Contents**

The contents of the section header fields for the loader section are:

<table>
<thead>
<tr>
<th>Name</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>s_name</td>
<td>.loader</td>
</tr>
<tr>
<td>s_paddr</td>
<td>0</td>
</tr>
<tr>
<td>s_vaddr</td>
<td>0</td>
</tr>
<tr>
<td>s_size</td>
<td>The size (in bytes) of the loader section</td>
</tr>
<tr>
<td>s_scnptr</td>
<td>Offset from the beginning of the XCOFF file to the first byte of the loader section data</td>
</tr>
<tr>
<td>s_relptr</td>
<td>0</td>
</tr>
<tr>
<td>s_lnnoptr</td>
<td>0</td>
</tr>
<tr>
<td>s_nreloc</td>
<td>0</td>
</tr>
<tr>
<td>s_nlnno</td>
<td>0</td>
</tr>
<tr>
<td>s_flags</td>
<td>STYP_LOADER</td>
</tr>
</tbody>
</table>

For general information on the XCOFF file format, see "XCOFF Object File Format."
Debug Section
The debug section contains the symbolic debugger stabstrings (symbol table strings). It is generated by the compilers and assemblers. It provides symbol attribute information for use by the symbolic debugger. The debug section has a section type flag of STYP_DEBUG in the XCOFF section header. By convention, .debug is the debug section name. The data in this section is referenced from entries in the XCOFF symbol table. A stabstring is a null-terminated character string. Each string is preceded by a 2-byte length field in XCOFF32 or a 4-byte length field in XCOFF64.

Field Definitions
The following two fields are repeated for each symbolic debugger stabstring:
- A 2-byte (XCOFF32) or 4-byte (XCOFF64) length field containing the length of the string. The value contained in the length field includes the length of the terminating null character but does not include the length of the length field itself.
- The symbolic debugger stabstring.

Refer to discussion of symbolic debugger stabstring grammar for the specific format of the stabstrings.

Debug Section Header Contents
The contents of the section header fields for the debug section are:

<table>
<thead>
<tr>
<th>Name</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>s_name</td>
<td>.debug</td>
</tr>
<tr>
<td>s_paddr</td>
<td>0</td>
</tr>
<tr>
<td>s_vaddr</td>
<td>0</td>
</tr>
<tr>
<td>s_size</td>
<td>The size (in bytes) of the debug section</td>
</tr>
<tr>
<td>s_scnpdr</td>
<td>Offset from the beginning of the XCOFF file to the first byte of the debug section data</td>
</tr>
<tr>
<td>s_relptr</td>
<td>0</td>
</tr>
<tr>
<td>s_linnptr</td>
<td>0</td>
</tr>
<tr>
<td>s_nreloc</td>
<td>0</td>
</tr>
<tr>
<td>s_nlnno</td>
<td>0</td>
</tr>
<tr>
<td>s_flags</td>
<td>STYP_DEBUG</td>
</tr>
</tbody>
</table>

For general information on the XCOFF file format, see "XCOFF Object File Format."

For more information on XCOFF file sections, see "Sections and Section Headers," "Debug Section," "Type-Check Section," "Exception Section," and "Comment Section."

Type-Check Section
The type-check section contains the type-checking hash strings and is produced by compilers and assemblers. It is used by the binder to detect variable mismatches and argument interface errors when linking separately compiled object files. (The type-checking hash strings in the loader section are used to detect these errors prior to running a program.) The type-check section has a section type flag of STYP_TYPCHK in the XCOFF section header. By convention, .typchk is the type-check section name. The strings in this section are referenced from entries in the XCOFF symbol table.

Field Definitions
The following two fields are repeated for each parameter type-checking string:
- A 2-byte length field containing the length of the type-checking string. The value contained in the length field does not include the length of the length field itself.
- The parameter type-checking hash string.
Type Encoding and Checking Format for Data

The type-checking hash strings are used to detect errors prior to execution of a program. Information about all external symbols (data and functions) is encoded by the compilers and then checked for consistency at bind time and load time. The type-checking strings are designed to enforce the maximum checking required by the semantics of each particular language supported, as well as provide protection to applications written in more than one language.

The type encoding and checking mechanism features 4-part hash encoding that provides some flexibility in checking. The mechanism also uses a unique value, UNIVERSAL, that matches any code. The UNIVERSAL hash can be used as an escape mechanism for assembly programs or for programs in which type information or subroutine interfaces might not be known. The UNIVERSAL hash is four blank ASCII characters (0x20202020) or four null characters (0x00000000).

The following fields are associated with the type encoding and checking mechanism:

- **code length**: A 2-byte field containing the length of the hash. This field has a value of 10.
- **language identifier**: A 2-byte code representing each language. These codes are the same as those defined for the e_lang field in the Exception section information.
- **general hash**: A 4-byte field representing the most general form by which a data symbol or function can be described. This form is the most common to languages supported by . If the information is incomplete or unavailable, a universal hash should be generated. The general hash is language-independent and must match for the binding to succeed.
- **language hash**: A 4-byte field containing a more detailed, language-specific representation of what is in the general hash. It allows for the strictest type-checking required by a given language. This part is used in intra-language binding and is not checked unless both symbols have the same language identifier.

Section Header Contents

The contents of the section header fields for the type-check section are:

<table>
<thead>
<tr>
<th>Name</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>s_name</td>
<td>.typchk</td>
</tr>
<tr>
<td>s_paddr</td>
<td>0</td>
</tr>
<tr>
<td>s_vaddr</td>
<td>0</td>
</tr>
<tr>
<td>s_size</td>
<td>The size (in bytes) of the type-check section</td>
</tr>
<tr>
<td>s_scnptr</td>
<td>Offset from the beginning of the XCOFF file to the first byte of the type-check section data</td>
</tr>
<tr>
<td>s_relptr</td>
<td>0</td>
</tr>
<tr>
<td>s_lnnptr</td>
<td>0</td>
</tr>
<tr>
<td>s_nreloc</td>
<td>0</td>
</tr>
<tr>
<td>s_nlnno</td>
<td>0</td>
</tr>
<tr>
<td>s_flags</td>
<td>STYP_TYPCHK.</td>
</tr>
</tbody>
</table>

For general information on the XCOFF file format, see "XCOFF Object File Format."

For more information on XCOFF file sections, see "Sections and Section Headers," "Debug Section," "Type-Check Section," "Exception Section," and "Comment Section."

Exception Section

The exception section contains addresses of trap instructions, source language identification codes, and trap reason codes. This section is produced by compilers and assemblers, and used during or after run time to identify the reason that a specific trap or exception occurred. The exception section has a section type flag of STYP_EXCEPT in the XCOFF section header. By convention, .except is the exception section name. Data in the exception section is referenced from entries in the XCOFF symbol table.
An exception table entry with a value of 0 in the e_reason field contains the symbol table index to a function's C_EXT, C_WEAKEXT, or C_HIDEXT symbol table entry. Reference from the symbol table to an entry in the exception table is via the function auxiliary symbol table entry. For more information on this entry, see "csect Auxiliary Entry for C_EXT, C_WEAKEXT and C_HIDEXT Symbols."

The C language structure for the exception section entries can be found in the exceptab.h file.

The exception section entries contain the fields shown in the following tables.

**Table 22. Initial Entry: Exception Section Structure**

<table>
<thead>
<tr>
<th>Field Name and Description</th>
<th>XCOFF32</th>
<th>XCOFF64</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>e_addr.e_symndx</code> Symbol table index for function</td>
<td>• Offset: 0</td>
<td>• Offset: 0</td>
</tr>
<tr>
<td></td>
<td>• Length: 4</td>
<td>• Length: 4</td>
</tr>
<tr>
<td><code>e_lang</code> Compiler language ID code</td>
<td>• Offset: 4</td>
<td>• Offset: 8</td>
</tr>
<tr>
<td></td>
<td>• Length: 1</td>
<td>• Length: 1</td>
</tr>
<tr>
<td><code>e_reason</code> Value 0 (exception reason code 0)</td>
<td>• Offset: 5</td>
<td>• Offset: 9</td>
</tr>
<tr>
<td></td>
<td>• Length: 1</td>
<td>• Length: 1</td>
</tr>
</tbody>
</table>

+Use “32” or “64” suffix when __XCOFF_HYBRID__ is defined. With e_addr.e_symndx, the suffix is added to e_addr (i.e. e_addr32.e_symndx).

**Table 23. Subsequent Entry: Exception Section Structure**

<table>
<thead>
<tr>
<th>Field Name and Description</th>
<th>XCOFF32</th>
<th>XCOFF64</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>e_addr.e_paddr</code> Address of the trap instruction</td>
<td>• Offset: 0</td>
<td>• Offset: 0</td>
</tr>
<tr>
<td></td>
<td>• Length: 4</td>
<td>• Length: 8</td>
</tr>
<tr>
<td><code>e_lang</code> Compiler language ID code</td>
<td>• Offset: 4</td>
<td>• Offset: 8</td>
</tr>
<tr>
<td></td>
<td>• Length: 1</td>
<td>• Length: 1</td>
</tr>
<tr>
<td><code>e_reason</code> Trap exception reason code</td>
<td>• Offset: 5</td>
<td>• Offset: 9</td>
</tr>
<tr>
<td></td>
<td>• Length: 1</td>
<td>• Length: 1</td>
</tr>
</tbody>
</table>

+Use “32” or “64” suffix when __XCOFF_HYBRID__ is defined. With e_addr.e_paddr, the suffix is added to e_addr (i.e. e_addr32.e_paddr).

**Field Definitions**

The following defines the fields listed of the exception section:

- `e_symndx` Contains an integer (overlaps the e_paddr field). When the e_reason field is 0, this field is the symbol table index of the function.
- `e_paddr` Contains a virtual address (overlaps the e_symndx field). When the e_reason field is nonzero, this field is the virtual address of the trap instruction.
Specifies the source language. The following list defines the possible values of the e_lang field.

<table>
<thead>
<tr>
<th>ID</th>
<th>Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>C</td>
</tr>
<tr>
<td>0x01</td>
<td>FORTRAN</td>
</tr>
<tr>
<td>0x02</td>
<td>Pascal</td>
</tr>
<tr>
<td>0x03</td>
<td>Ada</td>
</tr>
<tr>
<td>0x04</td>
<td>PL/I</td>
</tr>
<tr>
<td>0x05</td>
<td>BASIC</td>
</tr>
<tr>
<td>0x06</td>
<td>Lisp</td>
</tr>
<tr>
<td>0x07</td>
<td>COBOL</td>
</tr>
<tr>
<td>0x08</td>
<td>Modula2</td>
</tr>
<tr>
<td>0x09</td>
<td>C++</td>
</tr>
<tr>
<td>0x0A</td>
<td>RPG</td>
</tr>
<tr>
<td>0x0B</td>
<td>PL8, PLIX</td>
</tr>
<tr>
<td>0x0C</td>
<td>Assembly</td>
</tr>
<tr>
<td>0x0D-0xFF</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Specifies an 8-bit, compiler-dependent trap exception reason code. Zero is not a valid trap exception reason code because it indicates the start of exception table entries for a new function.

### Section Header Contents

The following fields are the contents of the section header fields for the exception section.

<table>
<thead>
<tr>
<th>Name</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>s_name</td>
<td>.except</td>
</tr>
<tr>
<td>s_paddr</td>
<td>0</td>
</tr>
<tr>
<td>s_vaddr</td>
<td>0</td>
</tr>
<tr>
<td>s_size</td>
<td>The size (in bytes) of the exception section</td>
</tr>
<tr>
<td>s_scnptr</td>
<td>Offset from the beginning of the XCOFF file to the first byte of the exception section data</td>
</tr>
<tr>
<td>s_relptr</td>
<td>0</td>
</tr>
<tr>
<td>s_lnnoptr</td>
<td>0</td>
</tr>
<tr>
<td>s_nreloc</td>
<td>0</td>
</tr>
<tr>
<td>s_nlnno</td>
<td>0</td>
</tr>
<tr>
<td>s_flags</td>
<td>STYP_EXCEPT</td>
</tr>
</tbody>
</table>

For general information on the XCOFF file format, see ["XCOFF Object File Format."](#)

For more information on XCOFF file sections, see ["Sections and Section Headers," "Debug Section," "Type-Check Section," "Exception Section," and "Comment Section."](#)

### Comment Section

The comment section contains information of special processing significance to an application. This section can be produced by compilers and assemblers and used during or after run time to fulfill a special processing need of an application. The comment section has a section type flag of STYP_INFO in the XCOFF section header. By convention, .info is the comment section name. Data in the comment section is referenced from C_INFO entries in the XCOFF symbol table.
The contents of a comment section consists of repeated instances of a 4-byte length field followed by a string of bytes (containing any binary value). The length of each string is stored in its preceding 4-byte length field. The string of bytes need not be terminated by a null character nor by any other special character. The specified length does not include the length of the length field itself. A length of 0 is allowed. The format of the string of bytes is not specified.

A comment section string is referenced from an entry in the XCOFF symbol table. The storage class of the symbol making a reference is C_INFO. See "Symbol Table Field Contents by Storage Class" for more information.

A C_INFO symbol is associated with the nearest C_FILE, C_EXT, C_WEAKEXT, or C_HIDEXT symbol preceding it.

Section Header Contents

The following fields are the contents of the section header fields for the comment section.

<table>
<thead>
<tr>
<th>Name</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>s_name</td>
<td>.info</td>
</tr>
<tr>
<td>s_paddr</td>
<td>0</td>
</tr>
<tr>
<td>s_vaddr</td>
<td>0</td>
</tr>
<tr>
<td>s_size</td>
<td>The size (in bytes) of the comment section</td>
</tr>
<tr>
<td>s_scnptr</td>
<td>Offset from the beginning of the XCOFF file to the first byte of the comment section data</td>
</tr>
<tr>
<td>s_relptr</td>
<td>0</td>
</tr>
<tr>
<td>s_lnnptr</td>
<td>0</td>
</tr>
<tr>
<td>s_nreloc</td>
<td>0</td>
</tr>
<tr>
<td>s_nlnno</td>
<td>0</td>
</tr>
<tr>
<td>s_flags</td>
<td>STYP_INFO</td>
</tr>
</tbody>
</table>

For general information on the XCOFF file format, see "XCOFF Object File Format."

For more information on XCOFF file sections, see "Sections and Section Headers," "Debug Section," "Type-Check Section," "Exception Section," and "Comment Section."

Relocation Information for XCOFF File (reloc.h)

The .text section and .data section may have relocation information. The relocation information is used by the binder to modify the .text section and .data section contents with address and byte-offset information of individual XCOFF object files collected into an XCOFF executable file.

The compilers and assemblers are responsible for generating the relocation entries for the .text and .data sections.

The binder generates relocation information for the .loader section, as required by the system loader.

Each relocation entry of the .text and .data section is 10 bytes long (14 for XCOFF64). (A relocation entry in the .loader section is 12 bytes long (16 for XCOFF64) and is explained in the loader section description in this document. See "Relocation Table Field Definitions" for more information.) The C language structure for a relocation entry can be found in the reloc.h file. A relocation entry contains the fields shown in the following table.

<table>
<thead>
<tr>
<th>Field Name and Description</th>
<th>XCOFF32</th>
<th>XCOFF64</th>
</tr>
</thead>
<tbody>
<tr>
<td>r_vaddr*</td>
<td>• Offset: 0</td>
<td>• Offset: 0</td>
</tr>
<tr>
<td></td>
<td>• Length: 4</td>
<td>• Length: 8</td>
</tr>
</tbody>
</table>

Table 24. Relocation Entry Structure

For general information on the XCOFF file format, see "XCOFF Object File Format."

For more information on XCOFF file sections, see "Sections and Section Headers," "Debug Section," "Type-Check Section," "Exception Section," and "Comment Section."

Relocation Information for XCOFF File (reloc.h)

The .text section and .data section may have relocation information. The relocation information is used by the binder to modify the .text section and .data section contents with address and byte-offset information of individual XCOFF object files collected into an XCOFF executable file.

The compilers and assemblers are responsible for generating the relocation entries for the .text and .data sections.

The binder generates relocation information for the .loader section, as required by the system loader.

Each relocation entry of the .text and .data section is 10 bytes long (14 for XCOFF64). (A relocation entry in the .loader section is 12 bytes long (16 for XCOFF64) and is explained in the loader section description in this document. See "Relocation Table Field Definitions" for more information.) The C language structure for a relocation entry can be found in the reloc.h file. A relocation entry contains the fields shown in the following table.

<table>
<thead>
<tr>
<th>Field Name and Description</th>
<th>XCOFF32</th>
<th>XCOFF64</th>
</tr>
</thead>
<tbody>
<tr>
<td>r_vaddr*</td>
<td>• Offset: 0</td>
<td>• Offset: 0</td>
</tr>
<tr>
<td></td>
<td>• Length: 4</td>
<td>• Length: 8</td>
</tr>
</tbody>
</table>

Table 24. Relocation Entry Structure

For general information on the XCOFF file format, see "XCOFF Object File Format."

For more information on XCOFF file sections, see "Sections and Section Headers," "Debug Section," "Type-Check Section," "Exception Section," and "Comment Section."

Relocation Information for XCOFF File (reloc.h)

The .text section and .data section may have relocation information. The relocation information is used by the binder to modify the .text section and .data section contents with address and byte-offset information of individual XCOFF object files collected into an XCOFF executable file.

The compilers and assemblers are responsible for generating the relocation entries for the .text and .data sections.

The binder generates relocation information for the .loader section, as required by the system loader.

Each relocation entry of the .text and .data section is 10 bytes long (14 for XCOFF64). (A relocation entry in the .loader section is 12 bytes long (16 for XCOFF64) and is explained in the loader section description in this document. See "Relocation Table Field Definitions" for more information.) The C language structure for a relocation entry can be found in the reloc.h file. A relocation entry contains the fields shown in the following table.

<table>
<thead>
<tr>
<th>Field Name and Description</th>
<th>XCOFF32</th>
<th>XCOFF64</th>
</tr>
</thead>
<tbody>
<tr>
<td>r_vaddr*</td>
<td>• Offset: 0</td>
<td>• Offset: 0</td>
</tr>
<tr>
<td></td>
<td>• Length: 4</td>
<td>• Length: 8</td>
</tr>
</tbody>
</table>

Table 24. Relocation Entry Structure

For general information on the XCOFF file format, see "XCOFF Object File Format."

For more information on XCOFF file sections, see "Sections and Section Headers," "Debug Section," "Type-Check Section," "Exception Section," and "Comment Section."
Table 24. Relocation Entry Structure (continued)

<table>
<thead>
<tr>
<th>Field Name and Description</th>
<th>XCOFF32</th>
<th>XCOFF64</th>
</tr>
</thead>
<tbody>
<tr>
<td>r_symndx</td>
<td>• Offset: 4</td>
<td>• Offset: 8</td>
</tr>
<tr>
<td>Symbol table index of item that is referenced</td>
<td>• Length: 4</td>
<td>• Length: 4</td>
</tr>
<tr>
<td>r_rsize</td>
<td>• Offset: 8</td>
<td>• Offset: 12</td>
</tr>
<tr>
<td>Relocation size and information</td>
<td>• Length: 1</td>
<td>• Length: 1</td>
</tr>
<tr>
<td>r_rtype</td>
<td>• Offset: 9</td>
<td>• Offset: 13</td>
</tr>
<tr>
<td>Relocation type</td>
<td>• Length: 1</td>
<td>• Length: 1</td>
</tr>
</tbody>
</table>

*Use "32" or "64" suffix when __XCOFF_HYBRID__ is defined.

The relocation entries for the .text and .data sections are part of their respective sections. The relocation entry refers to a location to be modified. The relocation entries for a section must be in ascending address order.

(The loader section contains a single set of relocation entries used by the system loader, so a section number is required within each relocation entry to identify the section that needs to be modified.)

Field Definitions
The following defines the relocation-information fields:

- r_vaddr: Specifies the virtual address of the value that requires modification by the binder. The byte offset value to the data that requires modification from the beginning of the section that contains the data can be calculated as follows:

  \[
  \text{offset in section} = r_{vaddr} - s_{paddr}
  \]

- r_symndx: Specifies a zero-based index into the XCOFF symbol table for locating the referenced symbol. The symbol table entry contains an address used to calculate a modification value to be applied at the r_vaddr relocation address.

- r_rsize: Specifies the relocation size and sign. Its contents are detailed in the following list:
  - 0x80 (1 bit) Indicates whether the relocation reference is signed (1) or unsigned (0).
  - 0x40 (1 bit) If this field is one, it indicates that the binder replaced the original instruction by a branch instruction to a special fixup instruction sequence.
  - 0x3F(6 bits) Specifies the bit length of the relocatable reference minus one. The current architecture allows for fields of up to 32 bits (XCOFF32) or 64 bits (XCOFF64) to be relocated.
Specifies an 8-bit relocation type field that indicates to the binder which relocation algorithm to use for calculating the modification value. This value is applied at the relocatable reference location specified by the \texttt{r_vaddr} field. The following relocation types are defined:

\begin{verbatim}
0x00 \ R\_POS
\end{verbatim}

Specifies positive relocation. Provides the address of the symbol specified by the \texttt{r_symndx} field.

\begin{verbatim}
0x01 \ R\_NEG
\end{verbatim}

Specifies negative relocation. Provides the negative of the address of the symbol specified by the \texttt{r_symndx} field.

\begin{verbatim}
0x02 \ R\_REL
\end{verbatim}

Specifies relative-to-self relocation. Provides a displacement value between the address of the symbol specified by the \texttt{r_symndx} field and the address of the csect to be modified.

\begin{verbatim}
0x03 \ R\_TOC
\end{verbatim}

Specifies relative-to-TOC relocation. Provides a displacement value that is the difference between the address value in the symbol specified by the \texttt{r_symndx} field and the address of the TOC anchor csect. The TOC anchor csect has a symbol table csect auxiliary entry with an \texttt{x_smclass} (storage mapping class) value of \texttt{XMC\_TC0}. The TOC anchor csect must be of zero length. There may be only one TOC anchor csect per XCOFF section.

\begin{verbatim}
0x04 \ R\_TRL
\end{verbatim}

Specifies TOC Relative Indirect Load (modifiable) relocation. Provides a displacement value that is the difference between the address value in the symbol specified by the \texttt{r_symndx} field and the address of the TOC anchor csect. This relocation entry is treated the same as an \texttt{R\_TOC} relocation entry. It provides the following additional information concerning the instruction being relocated: The instruction that is referenced by the \texttt{r_vaddr} field is a load instruction. That load instruction is permitted to be modified by the binder to become a compute address instruction. Changing an instruction from a load instruction to a compute address instruction avoids a storage reference during execution. A compute address instruction can be used if the address contained at the address specified by the \texttt{r_symndx} field has a value that itself references a \texttt{r_symndx} field that can be accessed with a valid in-range displacement relative to the TOC anchor address. That is, the target of the TOC entry is from -32,768 to 32,767, inclusive, from the TOC anchor address. If a compute address instruction is generated by the binder, the \texttt{R\_TRL} relocation type is changed to become a \texttt{R\_TRLA} type. This allows the reverse transformation, if required. Compilers are permitted to generate this relocation type.

\begin{verbatim}
0x13 \ R\_TRLA
\end{verbatim}

Specifies TOC Relative Load Address (modifiable LA to L) relocation. Provides a displacement value that is the difference between the address value in the symbol specified by the \texttt{r_symndx} field and the address of the TOC anchor csect. This relocation entry is treated the same as an \texttt{R\_TOC} relocation entry. It provides the following additional information concerning the instruction being relocated: The instruction that is referenced by the \texttt{r_vaddr} field is a compute address instruction. The compute address instruction is modified by the binder to become a load instruction whenever the calculated displacement value is outside the valid displacement range relative to the TOC anchor address. This relocation type provides the binder with a means to transform a compute address instruction into a load instruction whenever required. If a load instruction is generated by the binder, the \texttt{R\_TRLA} relocation type is changed to become an \texttt{R\_TRL} type. Compilers are not permitted to generate this relocation type.
Specifies Global Linkage-External TOC address relocation. Provides the address of the TOC associated with a defined external symbol. The external symbol with the required TOC address is specified by the \( r_{symndx} \) field of the relocation entry. This relocation entry provides a method of accessing the address of the TOC contained within the same executable where the \( r_{symndx} \) external symbol is defined.

Specifies local object TOC address relocation. Provides the address of the TOC associated with a defined external symbol. The external symbol for which the TOC address is required is specified by the \( r_{symndx} \) field of the relocation entry. The external symbol is defined locally within the resultant executable. This relocation entry provides a method of accessing the address of the TOC contained within the same executable where the \( r_{symndx} \) external symbol is defined.

Treated the same as the \( R_{POS} \) relocation type.

Treated the same as the \( R_{POS} \) relocation type.

Treated the same as the \( R_{POS} \) relocation type.

Treated the same as the \( R_{RBA} \) relocation type.

Specifies branch absolute relocation. Provides the address of the symbol specified by the \( r_{symndx} \) field as the target address of a branch instruction. The instruction can be modified to a (relative) branch instruction if the target address is relocatable.

Treated the same as the \( R_{RBR} \) relocation type.

Specifies (relative) branch relocation. Provides a displacement value between the address of the symbol specified by the \( r_{symndx} \) field and the address of the csect containing the branch instruction to be modified. The instruction can be modified to an absolute branch instruction if the target address is not relocatable.

The \( R_{RBR} \) relocation type is the standard branch relocation type used by compilers and assemblers for the . This relocation type along with \texttt{glink} code allows an executable object file to have a text section that is position-independent.
r_type continued

0x20 R_TLS
Specifies thread-local storage relocation, using the general-dynamic model. Provides an
offset into the thread-local storage for the module.

0x21 R_TLS_IE
Same as R_TLS, except that the initial-exec model is used. That is, the referenced
symbol must be exported by the main program or a module that is loaded at exec time.

0x22 R_TLS_LD
Same as R_TLS, except that the local-dynamic model is used. That is, the referenced
symbol must be in the referencing module.

0x23 R_TLS_LE
Same as R_TLS, except that local-exec model is used. That is, both the reference and
the referenced symbol must be in the main program.

0x24 R_TLSM
Specifies thread-local storage relocation. Provides a handle for the thread-local storage
of the referenced variable. The handle is used by the pthread runtime to locate the
thread-local storage.

0x25 R_TLSML
Specifies thread-local storage relocation. Provides a handle for the module containing the
reference. The r_symndx field must specify the symbol table index of the csect symbol
containing the reference.

0x30 R_TOCU
Specifies the high-order 16 bits of a TOC-relative relocation. Similar to the R_TOC
relocation, a displacement value is computed. The displacement value is the difference
between the address value in the symbol that the r_symndx field specifies and the
address of the TOC anchor csect. The high-order 16 bits of the displacement are used to
update the instruction. This relocation can overflow if the TOC size is larger than $2^{31}$
bytes.

0x31 R_TOCL
Specifies the low-order 16 bits of a TOC-relative relocation. Similar to the R_TOC
relocation, a displacement value is computed. The displacement value is the difference
between the address value in the symbol that the r_symndx field specifies and the
address of the TOC anchor csect. The low-order 16 bits of the displacement are used to
update the instruction.

### Additional Relocation Features

Standard practice is to retain relocation information only for unresolved references or references between
distinct sections. Once a reference is resolved, the relocation information is discarded. This is sufficient for
an incremental bind and a fixed address space model. To provide the capability for rebinding and handling a
relocatable address space model, the relocation information is not discarded from an XCOFF file.

For general information on the XCOFF file format, see "XCOFF Object File Format."

For more information on relocation field table definitions, see "Relocation Table Field Definitions" in the
loader section.

### Line Number Information for XCOFF File (linenum.h)

Line number entries are used by the symbolic debugger to debug code at the source level. When present,
there is a single line number entry for every source line that can have a symbolic debugger breakpoint.
The line numbers are grouped by function. The beginning of each function is identified by the _l_nlno field containing a value of 0. The first field, _l_symndx, is the symbol table index to the C_EXT, C_WEAKEXT, or C_HIDEEXT symbol table entry for the function.

Each line number entry is six bytes long. The C language structure for a line number entry can be found in the linenum.h file. A line number entry contains the fields shown in the following tables.

Table 25. Initial Line Number Structure Entry for Function

<table>
<thead>
<tr>
<th>Field Name and Description</th>
<th>XCOFF32</th>
<th>XCOFF64</th>
</tr>
</thead>
<tbody>
<tr>
<td>_l_addr._l_symndx*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Symbol table index for</td>
<td>Offset: 0</td>
<td>Offset: 0</td>
</tr>
<tr>
<td>function</td>
<td>Length: 4</td>
<td>Length: 4</td>
</tr>
<tr>
<td>_l_llno*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Value 0 (line number 0)</td>
<td>Offset: 4</td>
<td>Offset: 8</td>
</tr>
<tr>
<td></td>
<td>Length: 2</td>
<td>Length: 4</td>
</tr>
</tbody>
</table>

+Use “32” or “64” suffix when __XCOFF_HYBRID__ is defined. With _l_addr._l_symndx, the suffix is added to _l_addr (i.e. _l_addr32._l_symndx).

Table 26. Subsequent Line Number Entries for Function

<table>
<thead>
<tr>
<th>Field Name and Description</th>
<th>XCOFF32</th>
<th>XCOFF64</th>
</tr>
</thead>
<tbody>
<tr>
<td>_l_paddr*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Address at which break</td>
<td>Offset: 0</td>
<td>Offset: 0</td>
</tr>
<tr>
<td>point can be inserted</td>
<td>Length: 4</td>
<td>Length: 8</td>
</tr>
<tr>
<td>_l_llno*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Line number relative to</td>
<td>Offset: 4</td>
<td>Offset: 8</td>
</tr>
<tr>
<td>start of function</td>
<td>Length: 2</td>
<td>Length: 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

+Use “32” or “64” suffix when __XCOFF_HYBRID__ is defined. With _l_addr._l_paddr, the suffix is added to _l_addr (i.e. _l_addr32._l_paddr).

Field Definitions

The following list defines the line number entries:

- _l_symndx: Specifies the symbol table index to the function name (overlaps the _l_paddr field). When the _l_llno field is 0, this interpretation of the field is used.
- _l_paddr: Specifies the virtual address of the first instruction of the code associated with the line number (overlaps the _l_symndx field). When the _l_llno field is not 0, this interpretation of the field is used.
- _l_llno: Specifies either the line number relative to the start of a function or 0 to indicate the beginning of a function.

Note: If part of a function other than the beginning comes from an include file, the line numbers are absolute, rather than relative to the beginning of the function. (See the C_BINCL and C_EINCL symbol types in “Storage Classes by Usage and Symbol Value Classification” for more information.)

For general information on the XCOFF file format, see “XCOFF Object File Format.”

For information on debugging, see “Debug Section.”

Symbol Table Information

One composite symbol table is defined for an XCOFF file. The symbol table contains information required by both the binder (external symbols) and the symbolic debugger (function definitions and internal and external symbols).
The symbol table consists of a list of 18-byte, fixed-length entries. Each symbol represented in the symbol table consists of at least one fixed-length entry, and some are followed by auxiliary entries of the same size.

See the following information to learn more about the symbol table:

- Symbol Table Auxiliary Information
- Symbol Table Field Contents by Storage Class
- String Table

For each external symbol, one or more auxiliary entries are required that provide additional information concerning the external symbol. There are three major types of external symbols of interest to the binder, performing the following functions:

- Define replaceable units or csects.
- Define the external names for functions or entry points within csects.
- Reference the names of external functions in another XCOFF object.

For symbols defining a replaceable unit (csect), a csect auxiliary entry defines the length and storage-mapping class of the csect. For symbols defining external names for functions within a csect, the csect auxiliary entry points to the containing csect, the parameter type-checking information, and the symbolic debugger information for the function. For symbols referencing the name of an external function, a csect auxiliary entry identifies the symbol as an external reference and points to parameter type-checking information.

Symbol Table Contents
An XCOFF symbol table has the following general contents and ordering:

- The C_FILE symbol table entries used to bracket all the symbol table entries associated with a given source file.
- The C_INFO comment section symbol table entries that are of source file scope. These follow the C_FILE entry but before the first csect definition symbol table entry.
- The symbolic debugger symbol table entries that are of file scope. These follow the C_FILE entry but before the first csect entry.
- Csect definition symbol table entries used to define and bracket all the symbols contained with a csect.
- C_INFO comment section symbol table entries that follow a csect definition symbol table entry are associated with that csect.
- All symbolic debugger symbol table entries that follow a csect definition symbol table entry or label symbol table entry are associated with that csect or label.

The ordering of the symbol table must be arranged by the compilers and assemblers both to accommodate the symbolic debugger requirements and to permit effective management by the binder of the different sections of the object file as a result of such binder actions as garbage collection, incremental binding, and rebinding. This ordering is required by the binder so that if a csect is deleted or replaced, all the symbol table information associated with the csect can also be deleted or replaced. Likewise, if all the csects associated with a source file are deleted or replaced, all the symbol table and related information associated with the file can also be deleted or replaced.

Symbol Table Layout
The following example shows the general ordering of the symbol table.

```
un_external   Undefined global symbols

/file         Prolog --defines stabstring compaction level
/file         Source file 1
/info         Comment section reference symbol with file scope
/stab         Global Debug symbols of a file
csect        Replaceable unit definition (code)
```
Symbol Table Entry (syms.h): Each symbol, regardless of storage class and type, has a fixed-format entry in the symbol table. In addition, some symbol types may have additional (auxiliary) symbol table entries immediately following the fixed-format entry. Each entry in the symbol table is 18 bytes long. The C language structure for a symbol table entry can be found in the syms.h file. The index for the first entry in the symbol table is 0. The following table shows the structure of the fixed-format part of each symbol in the symbol table.

Table 27. Symbol Table Entry Format

<table>
<thead>
<tr>
<th>Field Name and Description</th>
<th>XCOFF32</th>
<th>XCOFF64</th>
</tr>
</thead>
<tbody>
<tr>
<td>n_name</td>
<td>• Offset: 0</td>
<td>• Offset: N/A</td>
</tr>
<tr>
<td></td>
<td>• Length: 8</td>
<td>• Length: N/A</td>
</tr>
<tr>
<td>n_zeroes</td>
<td>• Offset: 0</td>
<td>• Offset: N/A</td>
</tr>
<tr>
<td></td>
<td>• Length: 4</td>
<td>• Length: N/A</td>
</tr>
<tr>
<td>n_offset</td>
<td>• Offset: 4</td>
<td>• Offset: 8</td>
</tr>
<tr>
<td></td>
<td>• Length: 4</td>
<td>• Length: 8</td>
</tr>
<tr>
<td>n_value</td>
<td>• Offset: 8</td>
<td>• Offset: 0</td>
</tr>
<tr>
<td></td>
<td>• Length: 4</td>
<td>• Length: 8</td>
</tr>
<tr>
<td>n_scnum</td>
<td>• Offset: 12</td>
<td>• Offset: 12</td>
</tr>
<tr>
<td></td>
<td>• Length: 2</td>
<td>• Length: 2</td>
</tr>
</tbody>
</table>

.n.info Comment section reference symbol with csect scope

.function Local/External function
stab Debug and local symbols of function
.function Local/External function
stab Debug and local symbols of function

............
csect Replaceable unit definition (local statics)
stab Debug and local statics of file
............
csect Relocatable unit definition (global data)
external Defined global symbol
stab Debug info for global symbol
............
.pkstab Source file 2
pkstab Global Debug symbols of a file
csect Replaceable unit definition (code)
.pkstab Local/External function
pkstab Debug and local symbols of function
............
csect Replaceable unit definition (local statics)
stab Debug and Local statics of file
............
csect Replaceable unit definition (global data)
external Defined global symbol
pkstab Debug info for global symbol
.pkstab Source file

............
.Symbol Table Entry (syms.h): Each symbol, regardless of storage class and type, has a fixed-format entry in the symbol table. In addition, some symbol types may have additional (auxiliary) symbol table entries immediately following the fixed-format entry. Each entry in the symbol table is 18 bytes long. The C language structure for a symbol table entry can be found in the syms.h file. The index for the first entry in the symbol table is 0. The following table shows the structure of the fixed-format part of each symbol in the symbol table.

Table 27. Symbol Table Entry Format

<table>
<thead>
<tr>
<th>Field Name and Description</th>
<th>XCOFF32</th>
<th>XCOFF64</th>
</tr>
</thead>
<tbody>
<tr>
<td>n_name</td>
<td>• Offset: 0</td>
<td>• Offset: N/A</td>
</tr>
<tr>
<td></td>
<td>• Length: 8</td>
<td>• Length: N/A</td>
</tr>
<tr>
<td>n_zeroes</td>
<td>• Offset: 0</td>
<td>• Offset: N/A</td>
</tr>
<tr>
<td></td>
<td>• Length: 4</td>
<td>• Length: N/A</td>
</tr>
<tr>
<td>n_offset</td>
<td>• Offset: 4</td>
<td>• Offset: 8</td>
</tr>
<tr>
<td></td>
<td>• Length: 4</td>
<td>• Length: 8</td>
</tr>
<tr>
<td>n_value</td>
<td>• Offset: 8</td>
<td>• Offset: 0</td>
</tr>
<tr>
<td></td>
<td>• Length: 4</td>
<td>• Length: 8</td>
</tr>
<tr>
<td>n_scnum</td>
<td>• Offset: 12</td>
<td>• Offset: 12</td>
</tr>
<tr>
<td></td>
<td>• Length: 2</td>
<td>• Length: 2</td>
</tr>
</tbody>
</table>
### Field Name and Description

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
<th>XCOFF32</th>
<th>XCOFF64</th>
</tr>
</thead>
</table>
| n_type     | Basic and derived type specification | • Offset: 14  
• Length: 2 | • Offset: 14  
• Length: 2 |
| n_lang     | Source language ID (overlays first byte of n_type) | • Offset: 14  
• Length: 1 | • Offset: 14  
• Length: 1 |
| n_cpu      | CPU Type ID (overlays second byte of n_type) | • Offset: 15  
• Length: 1 | • Offset: 15  
• Length: 1 |
| n_sclass   | Storage class of symbol | • Offset: 16  
• Length: 1 | • Offset: 16  
• Length: 1 |
| n_numaux   | Number of auxiliary entries | • Offset: 17  
• Length: 1 | • Offset: 17  
• Length: 1 |

*Use “32” or “64” suffix when __XCOFF_HYBRID__ is defined.*

### Field Definitions:

The following defines the symbol table entry fields:

- **n_name**: Used by XCOFF32 only. Specifies an 8-byte, null-padded symbol name or symbolic debugger stabstring. The storage class field is used to determine if the field is a symbol name or symbolic debugger stabstring. By convention, a storage class value with the high-order bit on indicates that this field is a symbolic debugger stabstring.

  If the XCOFF32 symbol name is longer than 8 bytes, the field is interpreted as the following two fields:

  - **n_zeroes**: A value of 0 indicates that the symbol name is in the string table or .debug section (overlays first word of n_name).

  - **n_offset**: Specifies the byte offset to the symbol name in the string table or .debug section (overlays last 4 bytes of n_name). The byte offset is relative to the start of the string table or .debug section. A byte offset value of 0 is a null or zero-length symbol name.

- **n_offset**: For XCOFF64: Specifies the byte offset to the symbol name in the string table or .debug section. The byte offset is relative to the start of the string table or .debug section. A byte offset value of 0 is a null or zero-length symbol name. (For XCOFF32 only, used in conjunction with n_zeroes. See entry immediately above.)
n_value  Specifies the symbol value. The contents of the symbol value field is storage class-dependent, as shown in the following definitions:

<table>
<thead>
<tr>
<th>Content</th>
<th>Storage Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relocatable address</td>
<td>C_EXT, C_WEAKEXT, C_HIDEXT, C_FCN, C_BLOCK, C_STAT</td>
</tr>
<tr>
<td>Zero</td>
<td>C_GSYM, C_BCOMM, C_DECL, C_ENTRY, C_ESTAT, C_ECOMM</td>
</tr>
<tr>
<td>Offset in csect</td>
<td>C_FUN, C_STSYM</td>
</tr>
<tr>
<td>Offset in file</td>
<td>C_BINCL, C_EINCL</td>
</tr>
<tr>
<td>Offset in comment section</td>
<td>C_INFO</td>
</tr>
<tr>
<td>Symbol table index</td>
<td>C_FILE, C_BSTAT</td>
</tr>
<tr>
<td>Offset relative to stack frame</td>
<td>C_LSYM, C_PSYM</td>
</tr>
<tr>
<td>Register number</td>
<td>C_RPSYM, C_RSYM</td>
</tr>
<tr>
<td>Offset within common block</td>
<td>C_ECOML</td>
</tr>
</tbody>
</table>

n_scnum  Specifies a section number associated with one of the following symbols:

-2  Specifies N_DEBUG, a special symbolic debugging symbol.
-1  Specifies N_ABS, an absolute symbol. The symbol has a value but is not relocatable.
0  Specifies N_UNDEF, an undefined external symbol.

Any other value

Specifies the section number where the symbol was defined.

n_type  Used in COFF for type information. This use is obsolete in XCOFF. For C_EXT and C_HIDEXT symbols, this field should contain 0x0020 for function symbols and 0 otherwise. This field has a special purpose for C_FILE symbols. See "File Auxiliary Entry for the C_FILE Symbol" for more information.

n_sclass  Specifies the storage class of the symbol. The storclass.h and dbxstclass.h files contain the definitions of the storage classes. See "Symbol Table Field Contents by Storage Class" for more information.

n_numaux  Specifies the number of auxiliary entries for the symbol. In XCOFF64, auxiliary symbols have an identifying type field, but in XCOFF32, there is no type field. Therefore, if more than one auxiliary entry is required for a symbol, the order of the auxiliary entries is determined by convention.

For general information on the XCOFF file format, see "XCOFF Object File Format."

Symbol Table Auxiliary Information

The symbol table contains auxiliary entries to provide supplemental information for a symbol. The auxiliary entries for a symbol follow its symbol table entry. The length of each auxiliary entry is the same as a symbol table entry (18 bytes). The format and quantity of auxiliary entries depend on the storage class (n_sclass) and type (n_type) of the symbol table entry.

In XCOFF32, symbols having a storage class of C_EXT, C_WEAKEXT or C_HIDEXT and more than one auxiliary entry must have the csect auxiliary entry as the last auxiliary entry. In XCOFF64, the x_auxtype field of each auxiliary symbol table entry differentiates the symbols, but the convention is to generate the csect auxiliary symbol table entry last.
File Auxiliary Entry for C_FILE Symbols

The file auxiliary symbol table entry is defined to contain the source file name and compiler-related strings. A file auxiliary entry is optional and is used with a symbol table entry that has a storage-class value of C_FILE. The C language structure for a file auxiliary entry can be found in the x_file structure in the syms.h file.

The C_FILE symbol provides source file-name information, source-language ID and CPU-version ID information, and, optionally, compiler-version and time-stamp information.

The n_type field of the symbol table entry identifies the source language of the source file and the CPU version ID of the compiled object file. The field information is as follows:

<table>
<thead>
<tr>
<th>Source Language ID</th>
<th>Overlays the high-order byte of the n_type field. This field contains the source-language identifier. The values for this field are defined in the e_lang field in “Exception Section”. This field can be used by the symbolic debuggers to determine the source language.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The optional values for this field are 248 (TB_OBJECT) for symbols from object files with no C_FILE symbol table entry; or 249 (TB_FRONT) or 250 (TB_BACK) for generated entries used to provide debugging information. If the source language is TB_FRONT or TB_BACK, the 8-character name field begins with ' ' (blank), '0'(NULL). If the source language is TB_FRONT, the third byte is the stabstring compaction level for the object file, and the n_offset field contains the symbol table index of the TB_BACK symbol table entry, if it exists, or 0 otherwise.</td>
</tr>
</tbody>
</table>
CPU Version ID

Defined as the low-order byte of the n_type field. Describes the kind of instructions generated for the file. The following values are defined:

- **0**: Reserved.
- **1**: Specifies 32-bit mode.
- **2**: Reserved.
- **3**: Specifies the common intersection of 32-bit and Processor.
- **4**: Specifies Processor.
- **5**: Specifies any mix of instructions between different architectures.
- **6**: Specifies a mix of and instructions (\(\)).
- **7-223**: Reserved.
- **224**: Specifies instructions.
- **225-255**: Reserved.

If both fields are 0, no information is provided about the source language.

**File Name Auxiliary Entry Format**

<table>
<thead>
<tr>
<th>Offset</th>
<th>Length in Bytes</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>14</td>
<td>x_fname</td>
<td>Source file string</td>
</tr>
<tr>
<td>0</td>
<td>4</td>
<td>x_zeroes</td>
<td>Zero, indicating file string in string table (overlays first 4 bytes of x_fname)</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>x_offset</td>
<td>Offset of file string in string table (overlays 5th-8th bytes of x_fname)</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>x_ftype</td>
<td>File string type</td>
</tr>
<tr>
<td>15</td>
<td>2</td>
<td></td>
<td>Reserved. Must contain 0.</td>
</tr>
<tr>
<td>17</td>
<td>1</td>
<td>x_auxtype</td>
<td>Auxiliary symbol type(XCOFF64 only)</td>
</tr>
</tbody>
</table>
**Field Definitions:** The following defines the fields listed above:

- **x_fname**: Specifies the source file name or compiler-related string.
  
  If the file name or string is longer than 8 bytes, the field is interpreted as the following two fields:
  
  - **x_zeros**: A value of 0 indicates that the source file string is in the string table (overlays first 4 bytes of x_fname).
  
  - **x_offset**: Specifies the offset from the beginning of the string table to the first byte of the source file string (overlays last 4 bytes of x_fname).

- **x_ftype**: Specifies the source-file string type.
  
<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>XFT_FN</td>
</tr>
<tr>
<td></td>
<td>Specifies the source-file name</td>
</tr>
<tr>
<td>1</td>
<td>XFT_CT</td>
</tr>
<tr>
<td></td>
<td>Specifies the compiler time stamp</td>
</tr>
<tr>
<td>2</td>
<td>XFT_CV</td>
</tr>
<tr>
<td></td>
<td>Specifies the compiler version number</td>
</tr>
<tr>
<td>128</td>
<td>XFT_CD</td>
</tr>
<tr>
<td></td>
<td>Specifies compiler-defined information</td>
</tr>
</tbody>
</table>

- **x_zeroes**: A value of 0 indicates that the source file string is in the string table (overlays first 4 bytes of x_fname).

- **x_offset**: Specifies the offset from the beginning of the string table to the first byte of the source file string (overlays last 4 bytes of x_fname).

- **x_fnumaux**: (XCOFF64 only) Specifies the type of auxiliary entry. Contains _AUX_FILE for this auxiliary entry.

If the file auxiliary entry is not used, the symbol name is the name of the source file. If the auxiliary entry is used, then the symbol name should be .file and the first file auxiliary entry (by convention) contains the source file name. More than one file auxiliary entry is permitted for a given symbol table entry. The n_numaux field contains the number of file auxiliary entries.

**csect Auxiliary Entry for C_EXT, C_WEAKEXT, and C_HIDEXT Symbols**

The csect auxiliary entry identifies csects (section definitions), entry points (label definitions), and external references (label declarations). A csect auxiliary entry is required for each symbol table entry that has a storage class value of C_EXT, C_WEAKEXT, or C_HIDEXT. See "Symbol Table Entry (syms.h)" for more information. By convention, the csect auxiliary entry in an XCOFF32 file must be the last auxiliary entry for any external symbol that has more than one auxiliary entry. The C language structure for a csect auxiliary entry can be found in the x_csect structure in the syms.h file.

**Table 28. csect Auxiliary Entry Format**

<table>
<thead>
<tr>
<th>Field Name and Description</th>
<th>XCOFF32</th>
<th>XCOFF64</th>
</tr>
</thead>
</table>
| **x_scnlen**                               | • Offset: 0  
| (See field definition section)            | • Length: 4  
|                                             | • Offset: N/A  
|                                             | • Length: N/A |
| **x_scnlen_lo**                            | • Offset: N/A  
| (See field definition section)            | • Length: N/A  
| Low 4 bytes of section length             | • Offset: 0  
|                                             | • Length: 4  |
| **x_parmhash**                             | • Offset: 4  
| Offset of parameter type-check hash in .typchk section | • Length: 4  |
|                                             | • Offset: 4  
|                                             | • Length: 4  |
| **x_snhash**                               | • Offset: 8  
| .typchk section number                     | • Length: 2  |
|                                             | • Offset: 8  
|                                             | • Length: 2  |
Table 28. csect Auxiliary Entry Format (continued)

<table>
<thead>
<tr>
<th>Field Name and Description</th>
<th>XCOFF32</th>
<th>XCOFF64</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>x_smtyp</strong></td>
<td>Symbol alignment and type</td>
<td>• Offset: 10</td>
</tr>
<tr>
<td></td>
<td>3-bit symbol alignment (log 2)</td>
<td>• Length: 1</td>
</tr>
<tr>
<td></td>
<td>3-bit symbol type</td>
<td>• Offset: 10</td>
</tr>
<tr>
<td></td>
<td>• Length: 1</td>
<td>• Length: 1</td>
</tr>
<tr>
<td><strong>x_smclas</strong></td>
<td>Storage mapping class</td>
<td>• Offset: 11</td>
</tr>
<tr>
<td></td>
<td>• Length: 1</td>
<td>• Length: 1</td>
</tr>
<tr>
<td><strong>x_stab</strong></td>
<td>Reserved</td>
<td>• Offset: 12</td>
</tr>
<tr>
<td></td>
<td>• Length: 4</td>
<td>• Offset: N/A</td>
</tr>
<tr>
<td><strong>x_snstab</strong></td>
<td>Reserved</td>
<td>• Offset: 16</td>
</tr>
<tr>
<td></td>
<td>• Length: 2</td>
<td>• Length: N/A</td>
</tr>
<tr>
<td><strong>x_scnlen_hi</strong></td>
<td>(See field definition section)</td>
<td>• Offset: N/A</td>
</tr>
<tr>
<td></td>
<td>High 4 bytes of section</td>
<td>• Length: N/A</td>
</tr>
<tr>
<td></td>
<td>length</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Length: N/A</td>
<td>• Length: 4</td>
</tr>
<tr>
<td><strong>(pad)</strong></td>
<td>Reserved</td>
<td>• Offset: N/A</td>
</tr>
<tr>
<td></td>
<td>• Length: N/A</td>
<td>• Length: 1</td>
</tr>
<tr>
<td><strong>x_auxtype</strong></td>
<td>Contains _AUX_CSECT; indicates type of auxiliary entry</td>
<td>• Offset: N/A</td>
</tr>
<tr>
<td></td>
<td>• Length: N/A</td>
<td>• Offset: 17</td>
</tr>
<tr>
<td></td>
<td>• Length: 1</td>
<td>• Length: 1</td>
</tr>
</tbody>
</table>

Field Definitions: The following defines the fields listed above:

**x_scnlen** Specifies a meaning dependent on **x_smtyp** as follows:

<table>
<thead>
<tr>
<th>If</th>
<th>Then</th>
</tr>
</thead>
<tbody>
<tr>
<td>XTY_SD</td>
<td><strong>x_scnlen</strong> contains the csect length.</td>
</tr>
<tr>
<td>XTY_LD</td>
<td><strong>x_scnlen</strong> contains the symbol table index of the containing csect.</td>
</tr>
<tr>
<td>XTY_CM</td>
<td><strong>x_scnlen</strong> contains the csect length.</td>
</tr>
<tr>
<td>XTY_ER</td>
<td><strong>x_scnlen</strong> contains 0.</td>
</tr>
</tbody>
</table>

In the XCOFF64 format, the value of **x_scnlen** is divided into two fields: **x_scnlen_hi**, representing the upper 4 bytes of the value, and **x_scnlen_lo**, representing the lower 4 bytes of the value.

**x_parmhash** Specifies the byte offset of the parameter type-check string in the .typchk section. The byte offset is from the beginning of the .typchk section in an XCOFF file. The byte offset points to the first byte of the parameter type-check string (not to its length field). See "Type-Check Section" for more information. A value of 0 in the **x_parmhash** field indicates that the parameter type-checking string is not present for this symbol, and the symbol will be treated as having a universal hash. The value should be 0 for **C_HIDEXT** symbols.

**x_snhash** Specifies the .typchk section number. The XCOFF section number containing the parameter type-checking strings. The section numbers are one-based. For compatibility with object files generated by some compilers, if **x_parmhash** is not equal to 0 but **x_snhash** does equal 0, then the first .typchk section in the file is used. The value should be 0 for **C_HIDEXT** symbols.
x_smtyp  

Specifies symbol alignment and type:

**Bits 0-4**
Contains a 5-bit csect address alignment value (log base 2). For example, a value of 3 in this field indicates 23, or 8, meaning the csect is to be aligned on an 8-byte address value. The alignment value is used only when the value of bits 5-7 of the x_smtyp field is either XTY_SD or XTY_CM.

**Bits 5-7**
Contains a 3-bit symbol type field. See the definitions for bits 5-7 of the l_smtyp field in "Loader Section" for more information.
Specifies the csect storage-mapping class. This field permits the binder to arrange csects by their storage-mapping class. The `x_smclas` field is used only when the value of bits 5-7 of the `x_smtyp` field is either `XTY_SD` or `XTY_CM`.

The following storage-mapping classes are read-only and normally mapped to the `.text` section:

<table>
<thead>
<tr>
<th>Value</th>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>XMC_PR</td>
<td>Specifies program code. The csect contains the executable instructions of the program.</td>
</tr>
<tr>
<td>1</td>
<td>XMC_RO</td>
<td>Specifies a read-only constant. The csect contains data that is constant and will not change during execution of the program.</td>
</tr>
<tr>
<td>2</td>
<td>XMC_DB</td>
<td>Specifies the debug dictionary table. The csect contains symbolic-debugging data or exception-processing data. This storage mapping class was defined to permit compilers with special symbolic-debugging or exception-processing requirements to place data in csects that are loaded at execution time but that can be collected separately from the executable code of the program.</td>
</tr>
<tr>
<td>6</td>
<td>XMC_GL</td>
<td>Specifies global linkage. The csect provides the interface code necessary to handle csect relative calls to a target symbol that can be out-of-module. This global linkage csect has the same name as the target symbol and becomes the local target of the relative calls. As a result, the csect maintains position-independent code within the <code>.text</code> section of the executable XCOFF object file.</td>
</tr>
<tr>
<td>7</td>
<td>XMC_XO</td>
<td>Specifies extended operation. A csect of this type has no dependency on (references through) the TOC. It is intended to reside at a fixed address in memory such that it can be the target of a branch-absolute instruction.</td>
</tr>
<tr>
<td>12</td>
<td>XMC_TI</td>
<td>Reserved.</td>
</tr>
<tr>
<td>13</td>
<td>XMC_TB</td>
<td>Reserved.</td>
</tr>
</tbody>
</table>

The following storage-mapping classes are read/write and normally mapped to the `.data` or `.bss` section:

<table>
<thead>
<tr>
<th>Value</th>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>XMC_RW</td>
<td>Specifies read/write data. A csect of this type contains initialized or uninitialized data that is permitted to be modified during program execution. If the <code>x_smtyp</code> value is <code>XTY_SD</code>, the csect contains initialized data and is mapped into the <code>.data</code> section. If the <code>x_smtyp</code> value is <code>XTY_CM</code>, the csect is uninitialized and is mapped into the <code>.bss</code> section. Typically, all the initialized static data from a C source file is contained in a single csect of this type. The csect would have a storage class value of <code>C_HIDEXT</code>. An initialized definition for a global data scalar or structure from a C source file is contained in its own csect of this type. The csect would have a storage class value of <code>C_EXT</code>. A csect of this type is accessible by name references from other object files.</td>
</tr>
</tbody>
</table>
**Value Class**

**Description**

15 **XMC_TC0**

Specifies TOC anchor for TOC addressability. This is a zero-length csect whose n_value address provides the base address for TOC relative addressability. Only one csect of type **XMC_TC0** is permitted per section of an XCOFF object file. In implementations that permit compilers and assemblers to generate multiple .data sections, there must be a csect of type **XMC_TC0** in each section that contains data that is referenced (by way of a relocation entry) as a TOC-relative data item. Some hardware architectures limit the value that a relative displacement field within a load instruction may contain. This limit then becomes an inherent limit on the size of a TOC for an executable XCOFF object. For RS/6000, this limit is 65,536 bytes, or 16,384 4-byte TOC entries.

3 **XMC_TC**

22 **XMC_TE**

Specifies general TOC entries. Csects of this type are the same size as a pointer and contain the address of other csects or global symbols. These csects provide addressability to other csects or symbols. The symbols might be in either the local executable XCOFF object or another executable XCOFF object. The binder uses special processing semantics to eliminate duplicate TOC entries as follows:

- Symbols that have a storage class value of **C_EXT** are global symbols and must have names (a non-null n_name field). These symbols require no special TOC processing logic to combine duplicate entries. Duplicate entries with the same n_name value are combined into a single entry.

- Symbols that have a storage class value of **C_HIDEXT** are not global symbols, and duplicate entries are resolved by context. Any two such symbols are defined as duplicates and combined into a single entry whenever the following conditions are met:
  - The n_name fields are the same. That is, they have either a null name or the same name string.
  - Each is the same size as a pointer.
  - Each has a single RLD entry that references external symbols with the same name.

To minimize the number of duplicate TOC entries that the binder cannot combine, compilers and assemblers should adhere to a common naming convention for TOC entries. By convention, compilers and assemblers produce TOC entries that have a storage class value of **C_HIDEXT** and an n_name string that is the same as the n_name value for the symbol that the TOC entry addresses.

Storage-mapping classes **XMC_TC** and **XMC_TE** are equivalent, except that the binder should map **XMC_TE** symbols after **XMC_TC** and **XMC_TD** symbols.
<table>
<thead>
<tr>
<th>Value</th>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
</table>
| 16    | XMC_TD  | Specifies scalar data entry in the TOC. A csect that is a special form of an XMC_RW csect that is directly accessed from the TOC by compiler generated code. This lets some frequently used global symbols be accessed directly from the TOC rather than indirectly through an address pointer csect contained in the TOC. A csect of type XMC_TD has the following characteristics:  
  - The compiler generates code that is TOC relative to directly access the data contained in the csect of type XMC_TD.  
  - It is 4-bytes long or less.  
  - It has initialized data that can be modified as the program runs.  
  - If a same named csect of type XMC_RW or XMC_UA exist, it is replaced by the XMC_TD csect.|
| 10    | XMC_DS  | Specifies a csect containing a function descriptor, which contains the following three values:  
  - The address of the executable code for a function.  
  - The address of the TOC anchor (TOC base address) of the module that contains the function.  
  - The environment pointer (used by languages such as Pascal and PL/I).  
For the cases where TOC scalar cannot reside in the TOC, the binder must be capable of transforming the compiler generated TOC relative instruction into a conventional indirect addressing instruction sequence. This transformation is necessary if the TOC scalar is contained in a shared object.|
| 8     | XMC_SV  | Specifies 32-bit supervisor call descriptor csect. The supervisor call descriptors are contained within the operating system kernel. To an application program, the reference to a supervisor call descriptor is treated the same as a reference to a regular function descriptor. It is through the import/export mechanism that a function descriptor is treated as a supervisor call descriptor. These symbols are only available to 32-bit programs.|
| 17    | XMC_SV64| Specifies 64-bit supervisor call descriptor csect. See XMV_SV for supervisor call information. These symbols are only available to 64-bit programs.|
| 18    | XMC_SV3264| Specifies supervisor call descriptor csect for both 32-bit and 64-bit. See XMV_SV for supervisor call information. These symbols are available to both 32-bit and 64-bit programs.|
| 4     | XMC_UA  | Unclassified. This csect is treated as read/write. This csect is frequently produced by an assembler or object file translator program that cannot determine the true classification of the resultant csect.|
| 9     | XMC_BS  | Specifies BSS class (uninitialized static internal). A csect of this type is uninitialized, and is intended to be mapped into the .bss section. This type of csect must have a x_smtyp value of XTY_CM.|

Chapter 2. File Formats
**Value Class**

**Description**

11 XMC_UC

Specifies unnamed FORTRAN common. A csect of this type is intended for an unnamed and uninitialized FORTRAN common. It is intended to be mapped into the .bss section. This type of csect must have a \texttt{x_smtyp} value of \texttt{XTY_CM}.

The following storage mapping class is read-write and is mapped to the .tdata section:

20 XMC_TL

Specifies read/write thread-local data. A csect of this type contains initialized data that is local to every thread in a process. When a new thread is created, a csect with type \texttt{XMC_TL} is used to initialize the thread-local data for the thread.

The following storage mapping class is read-write and is mapped to the .tbss section:

21 XMC_UL

Specifies read/write thread-local data. A csect of this type contains uninitialized data that is local to every thread in a process. When a new thread is created, the thread-local storage for a csect of this type is initialized to zero.

\texttt{x_stab} Reserved (Unused for 64-bit).
\texttt{x_snstab} Reserved (Unused for 64-bit).

### Auxiliary Entries for the C\_EXT, C\_WEAKEXT, and C\_HIDEXT Symbols

Auxiliary symbol table entries are defined in XCOFF to contain reference and size information associated with a defined function. These auxiliary entries are produced by compilers and assembler for use by the symbolic debuggers. In XCOFF32, a function auxiliary symbol table entry contains the required information. In XCOFF64, both a function auxiliary entry and an exption auxiliary entry may be needed. When both auxiliary entries are generated for a single \texttt{C\_EXT, C\_WEAKEXT, or C\_HIDEXT} symbol, the \texttt{x_size} and \texttt{x_endndx} fields must have the same values.

The function auxiliary symbol table entry is defined in the following table.

*Table 29. Function Auxiliary Entry Format*

<table>
<thead>
<tr>
<th>Field Name and Description</th>
<th>XCOFF32</th>
<th>XCOFF64</th>
</tr>
</thead>
</table>
| \texttt{x_exptr}           | • Offset: 0  
|                            | • Length: 4  | • Offset: N/A  
|                            | • Length: N/A |
| \texttt{x_fsize}           | • Offset: 4  
|                            | • Length: 4  | • Offset: 8  
|                            | • Length: 4  |
| \texttt{x_lnnoptr}         | • Offset: 8  
|                            | • Length: 4  | • Offset: 0  
|                            | • Length: 8  |
| \texttt{x_endndx}          | • Offset: 12  
|                            | • Length: 4  | • Offset: 12  
|                            | • Length: 4  |
| \texttt{(pad)}             | • Offset: 16  
|                            | • Length: 1  | • Offset: 16  
|                            | • Length: 1  |
| \texttt{x_auxtype}         | • Offset: N/A  
|                            | • Length: N/A | • Offset: 17  
|                            | • Length: 1  |
Field Definitions: The following defines the fields listed in the Function Auxiliary Entry Format table:

**x_exptr**  (XCOFF32 only) This field is a file pointer to an exception table entry. The value is the byte offset from the beginning of the XCOFF object file. In an XCOFF64 file, the exception table offsets are in an exception auxiliary symbol table entry.

**x_fsize**  Specifies the size of the function in bytes.

**x_lnnopr**  Specifies a file pointer to the line number. The value is the byte offset from the beginning of the XCOFF object file.

**x_endndx**  Specifies the symbol table index of the next entry beyond this function.

The exception auxiliary symbol table entry, defined in XCOFF64 only, is shown in the following table.

**Table 30. Exception Auxiliary Entry Format (XCOFF64 only)**

<table>
<thead>
<tr>
<th>Offset</th>
<th>Length</th>
<th>Name and Description</th>
</tr>
</thead>
</table>
| 0      | 8      | **x_exptr**  
File offset to exception table entry. |
| 8      | 4      | **x_fsize**  
Size of function in bytes |
| 12     | 4      | **x_endndx**  
Symbol table index of next entry beyond this function |
| 16     | 1      | (pad)  Unused |
| 17     | 1      | **x_auxtype**  
Contains _AUX_EXCEPT; Type of auxiliary entry |

Field Definitions: The following defines the fields listed in the Exception Auxiliary Entry Format table:

**x_exptr**  This field is a file pointer to an exception table entry. The value is the byte offset from the beginning of the XCOFF object file.

**x_fsize**  Specifies the size of the function in bytes.

**x_endndx**  Specifies the symbol table index of the next entry beyond this function.

Block Auxiliary Entry for the C_BLOCK and C_FCN Symbols

The section auxiliary symbol table entry is defined in XCOFF to provide information associated with the begin and end blocks of functions. The section auxiliary symbol table entry is produced by compilers for use by the symbolic debuggers.

**Table 31. Table Entry Format**

<table>
<thead>
<tr>
<th>Field Name and Description</th>
<th>XCOFF32</th>
<th>XCOFF64</th>
</tr>
</thead>
</table>
| (no name)  
Reserved            | • Offset: 0  
• Length: 4             | • Offset: N/A  
• Length: N/A |
| x_lnnopr  
Source line number        | • Offset: 4  
• Length: 2             | • Offset: 0  
• Length: 4 |
| (no name)  
Reserved            | • Offset: 6  
• Length: 12            | • Offset: 4  
• Length: 13 |
| x_auxtype  
Contains _AUX_SYM; Type of auxiliary entry | • Offset: N/A  
• Length: N/A             | • Offset: 17  
• Length: 1 |
Field Definitions: The following defines the fields above:

(no name) Reserved.
x_linnno Specifies the line number of a source file. The maximum value of this field is 65535 for XCOFF64 and 2^{32} for XCOFF64.
(no name) Reserved.

Section Auxiliary Entry for the C_STAT Symbol
The section auxiliary symbol table entry ID is defined in XCOFF32 to provide information in the symbol table concerning the size of sections produced by a compiler or assembler. The generation of this information by a compiler is optional, and is ignored and removed by the binder.

Table 32. Section Auxiliary Entry Format (XCOFF32 Only)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Length in Bytes</th>
<th>Name and Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4</td>
<td>x_scnlen</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Section length</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>x_nreloc</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Number of relocation entries</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>x_nlinno</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Number of line numbers</td>
</tr>
<tr>
<td>8</td>
<td>10</td>
<td>(no name)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Field Definitions: The following list defines the fields:

x_scnlen Specifies section length in bytes.
x_nreloc Specifies the number of relocation entries. The maximum value of this field is 65535.
x_nlinno Specifies the number of line numbers. The maximum value of this field is 65535.
(no name) Reserved.

For general information on the XCOFF file format, see “XCOFF Object File Format.” For more information on the symbol table, see “Symbol Table Information.”

For information on debugging, see “Debug Section.”

Symbol Table Field Contents by Storage Class
This section defines the symbol table field contents for each of the defined storage classes (n_sclass) that are used in XCOFF. The following table lists storage class entries in alphabetic order. See “Symbol Table Entry (syms.h)” for more information.
### Table 33. Symbol Table by Storage Class

<table>
<thead>
<tr>
<th>Class Definition</th>
<th>Field Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>C_BCOMM 135</strong> Beginning of common block</td>
<td>n_name Name of the common block*</td>
</tr>
<tr>
<td></td>
<td>n_value 0, undefined</td>
</tr>
<tr>
<td></td>
<td>n_scnum N_DEBUG</td>
</tr>
<tr>
<td></td>
<td>Aux. Entry</td>
</tr>
<tr>
<td><strong>C_BINCL 108</strong> Beginning of include file</td>
<td>n_name Source name of the include file**</td>
</tr>
<tr>
<td></td>
<td>n_value File pointer</td>
</tr>
<tr>
<td></td>
<td>n_scnum N_DEBUG</td>
</tr>
<tr>
<td></td>
<td>Aux. Entry</td>
</tr>
<tr>
<td><strong>C_BLOCK 100</strong> Beginning or end of inner block</td>
<td>n_name bb or eb</td>
</tr>
<tr>
<td></td>
<td>n_value Relocatable address</td>
</tr>
<tr>
<td></td>
<td>n_scnum N_SCNUM</td>
</tr>
<tr>
<td></td>
<td>Aux. Entry BLOCK</td>
</tr>
<tr>
<td><strong>C_BSTAT 143</strong> Beginning of static block</td>
<td>n_name bs</td>
</tr>
<tr>
<td></td>
<td>n_value Symbol table index</td>
</tr>
<tr>
<td></td>
<td>n_scnum N_DEBUG</td>
</tr>
<tr>
<td></td>
<td>Aux. Entry</td>
</tr>
<tr>
<td><strong>CDECL 140</strong> Declaration of object (type)</td>
<td>n_name Debugger stabstring*</td>
</tr>
<tr>
<td></td>
<td>n_value 0, undefined</td>
</tr>
<tr>
<td></td>
<td>n_scnum N_SCNUM</td>
</tr>
<tr>
<td></td>
<td>Aux. Entry</td>
</tr>
<tr>
<td>Class Definition</td>
<td>Field Contents</td>
</tr>
<tr>
<td>------------------</td>
<td>----------------</td>
</tr>
</tbody>
</table>
| C_ECOML 136 Local member of common block | n_name  
  Debugger stabstring*  
  n_value  
  Offset within common block  
  n_scnum  
  N_ABS  
  Aux. Entry |
| C_ECOMM 137 End of common block | n_name  
  Debugger stabstring*  
  n_value  
  0, undefined  
  n_scnum  
  N_DEBUG  
  Aux. Entry |
| C_EINCL 109 End of include file | n_name  
  Source name of the include file**  
  n_value  
  File pointer  
  n_scnum  
  N_DEBUG  
  Aux. Entry |
| C_ENTRY 141 Alternate entry | n_name  
  *  
  n_value  
  0, undefined  
  n_scnum  
  N_DEBUG  
  Aux. Entry |
| C_ESTAT 144 End of static block | n_name  
  .es  
  n_value  
  0, undefined  
  n_scnum  
  N_DEBUG  
  Aux. Entry |
<table>
<thead>
<tr>
<th>Class Definition</th>
<th>Field Contents</th>
</tr>
</thead>
</table>
| C_EXT 2 External symbol (defining external symbols for binder processing) | n_name  
Symbol Name**  
n_value  
Relocatable address  
n_scnum  
N_SCNUM or N_UNDEF  
Aux. Entry  
FUNCTION CSECT |
| C_FCN 101 Beginning or end of function | n_name  
.bf or .ef  
n_value  
Relocatable address  
n_scnum  
N_SCNUM  
Aux. Entry  
BLOCK |
| C_FILE 103 Source file name and compiler information | n_name  
.file or source file name (if no auxiliary entries)**  
n_value  
Symbol table index  
n_scnum  
N_DEBUG  
Aux. Entry  
FILE |
| C_FUN 142 Function or procedure | n_name  
Debugger stabstring*  
n_value  
Offset within containing csect  
n_scnum  
N_ABS  
Aux. Entry |
| C_GSYM 128 Global variable | n_name  
Debugger stabstring*  
n_value  
0, undefined  
n_scnum  
N_DEBUG  
Aux. Entry |
<table>
<thead>
<tr>
<th>Class Definition</th>
<th>Field Contents</th>
</tr>
</thead>
</table>
| C_GTLS 145 Global thread-local variable | n_name: Debugger stabstring*  
| | n_value: 0, undefined  
| | n_scnum: N_DEBUG  
| | Aux. Entry |
| C_HIDEXT 107 Unnamed external symbol | n_name: Symbol Name or null**  
| | n_value: Relocatable address  
| | n_scnum: N_SCNUM  
| | Aux. Entry FUNCTION CSECT |
| C_INFO 100 Comment section reference | n_name: Info Name Identifier or null**  
| | n_value: Offset within comment section  
| | n_scnum: N_SCNUM  
| | Aux. Entry |
| C_LSYM 129 Automatic variable allocated on stack | n_name: Debugger stabstring*  
| | n_value: Offset relative to stack frame  
| | n_scnum: N_ABS  
| | Aux. Entry |
| C_NULL 0 Symbol table entry marked for deletion. | n_name: 0x00DE1E00  
| | n_value: Any  
<p>| | n_scnum: Aux. Entry |</p>
<table>
<thead>
<tr>
<th>Class Definition</th>
<th>Field Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>C_PSYM 130</strong></td>
<td><strong>Argument to subroutine allocated on stack</strong></td>
</tr>
<tr>
<td></td>
<td>n_name \n Debugger stabstring*</td>
</tr>
<tr>
<td></td>
<td>n_value \n Offset relative to stack frame</td>
</tr>
<tr>
<td></td>
<td>n_scnum \n N_ABS</td>
</tr>
<tr>
<td></td>
<td>Aux. Entry</td>
</tr>
<tr>
<td><strong>C_RPSYM 132</strong></td>
<td><strong>Argument to function or procedure stored in register</strong></td>
</tr>
<tr>
<td></td>
<td>n_name \n Debugger stabstring*</td>
</tr>
<tr>
<td></td>
<td>n_value \n Register number</td>
</tr>
<tr>
<td></td>
<td>n_scnum \n N_ABS</td>
</tr>
<tr>
<td></td>
<td>Aux. Entry</td>
</tr>
<tr>
<td><strong>C_RSYM 131</strong></td>
<td><strong>Register variable</strong></td>
</tr>
<tr>
<td></td>
<td>n_name \n Debugger stabstring*</td>
</tr>
<tr>
<td></td>
<td>n_value \n Register number</td>
</tr>
<tr>
<td></td>
<td>n_scnum \n N_ABS</td>
</tr>
<tr>
<td></td>
<td>Aux. Entry</td>
</tr>
<tr>
<td><strong>C_STAT 3</strong></td>
<td><strong>Static symbol (Unknown. Some compilers generate these symbols in the symbol table to identify size of the .text, .data, and .bss sections. Not used or preserved by binder.)</strong></td>
</tr>
<tr>
<td></td>
<td>n_name \n Symbol Name**</td>
</tr>
<tr>
<td></td>
<td>n_value \n Relocatable address</td>
</tr>
<tr>
<td></td>
<td>n_scnum \n N_SCNUM</td>
</tr>
<tr>
<td></td>
<td>Aux. Entry</td>
</tr>
<tr>
<td><strong>C_STSYM 133</strong></td>
<td><strong>Statically allocated symbol</strong></td>
</tr>
<tr>
<td></td>
<td>n_name \n Debugger stabstring*</td>
</tr>
<tr>
<td></td>
<td>n_value \n Offset within csect</td>
</tr>
<tr>
<td></td>
<td>n_scnum \n N_DEBUG</td>
</tr>
<tr>
<td></td>
<td>Aux. Entry</td>
</tr>
</tbody>
</table>
### Table 33. Symbol Table by Storage Class (continued)

<table>
<thead>
<tr>
<th>Class Definition</th>
<th>Field Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>C_STTLS</strong> 146 Static thread-local variable</td>
<td><code>n_name</code> Debugger stabstring*</td>
</tr>
<tr>
<td></td>
<td><code>n_value</code> 0, undefined</td>
</tr>
<tr>
<td></td>
<td><code>n_scnum</code> N_DEBUG</td>
</tr>
<tr>
<td></td>
<td>Aux. Entry</td>
</tr>
<tr>
<td><strong>C_TCSYM</strong> 134 Reserved</td>
<td><code>n_name</code> Debugger stabstring*</td>
</tr>
<tr>
<td></td>
<td><code>n_value</code> n_scnum</td>
</tr>
<tr>
<td></td>
<td>Aux. Entry</td>
</tr>
<tr>
<td><strong>C_WEAKEXT</strong> 111 Weak external symbol (defining weak external symbols for binder processing)</td>
<td><code>n_name</code> Symbol Name**</td>
</tr>
<tr>
<td></td>
<td><code>n_value</code> Relocatable address</td>
</tr>
<tr>
<td></td>
<td><code>n_scnum</code> N_SCNUM or N_UNDEF</td>
</tr>
<tr>
<td></td>
<td>Aux. Entry FUNCTION CSECT</td>
</tr>
</tbody>
</table>

### Notes:

1. *For long name, the n_offset value is an offset into the .debug section.*
2. **For long name, the n_offset value is an offset into the string table.*

### Storage Classes by Usage and Symbol Value Classification

Following are the storage classes used and relocated by the binder. The symbol values (`n_value`) are addresses.

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>C_EXT</strong></td>
<td>Specifies an external or global symbol</td>
</tr>
<tr>
<td><strong>C_WEAKEXT</strong></td>
<td>Specifies an external or global symbol with weak binding</td>
</tr>
<tr>
<td><strong>C_HIDEXT</strong></td>
<td>Specifies an internal symbol</td>
</tr>
<tr>
<td><strong>C_BLOCK</strong></td>
<td>Specifies the beginning or end of an inner block (.bb or .eb)</td>
</tr>
<tr>
<td><strong>C_FCN</strong></td>
<td>Specifies the beginning or end of a function (.bf or .ef only)</td>
</tr>
<tr>
<td><strong>C_STAT</strong></td>
<td>Specifies a static symbol (contained in statics csect)</td>
</tr>
</tbody>
</table>

Following are storage classes used by the binder and symbolic debugger or by other utilities for file scoping and accessing purposes:

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>C_FILE</strong></td>
<td>Specifies the source file name. The n_value field holds the symbol index of the next file entry. The n_name field is the name of the file.</td>
</tr>
<tr>
<td><strong>C_BINCL</strong></td>
<td>Specifies the beginning of include header file. The n_value field is the line number byte offset in the object file to the first line number from the include file.</td>
</tr>
</tbody>
</table>
C_EINCL  Specifies the end of include header file. The n_value field is the line number byte offset in the object file to last line number from the include file.

C_INFO   Specifies the location of a string in the comment section. The n_value field is the offset to a string of bytes in the specified STYP_INFO section. The string is preceded by a 4-byte length field. The n_name field is preserved by the binder. An application-defined unique name in this field can be used to filter access to only those comment section strings intended for the application.

Following are the storage classes that exist only for symbolic debugging purposes:

C_BCOMM  Specifies the beginning of a common block. The n_value field is meaningless; the name is the name of the common block.

C_ECOML  Specifies a local member of a common block. The n_value field is byte-offset within the common block.

C_BSTAT  Specifies the beginning of a static block. The n_value field is the symbol table index of the csect containing static symbols; the name is .bs.

C_ESTAT  Specifies the end of a static block. The n_value field is meaningless; the name is .es.

C_DECL   Specifies a declaration of object (type declarations). The n_value field is undefined.

C_ENTRY  Specifies an alternate entry (FORTRAN) and has a corresponding C_EXT or C_WEAKEXT symbol. The n_value field is undefined.

C_FUN    Specifies a function or procedure. May have a corresponding C_EXT or C_WEAKEXT symbol. The n_value field is byte-offset within the containing csect.

C_GSYM   Specifies a global variable and has a corresponding C_EXT or C_WEAKEXT symbol. The n_value field is undefined.

C_LSYM   Specifies an automatic variable allocated on the stack. The n_value field is byte offset relative to the stack frame (platform dependent).

C_PSYM   Specifies an argument to a subroutine allocated on the stack. The n_value field is byte-offset relative to the stack frame (platform dependent).

C_RSYM   Specifies a register variable. The n_value field is the register number.

C_RPSYM  Specifies an argument to a function or procedure stored in a register. The n_value field is the register number where argument is stored.

C_STSYM  Specifies a statically allocated symbol. The n_value field is byte-offset within csect pointed to by containing C_BSTAT entry.

C_GTLS   Specifies a global thread-local variable and follows a C_EXT or C_WEAKEXT symbol with the same name. The n_value field is undefined.

C_STTLS  Specifies a static thread-local variable and follows a C_HIDEXT symbol with the same name. The n_value field is undefined.

For general information on the XCOFF file format, see "XCOFF Object File Format." For more information on the symbol table, see "Symbol Table Information."

For information on debugging, see "Debug Section."

**String Table**

IN XCOFF32, the string table contains the names of symbols that are longer than 8 bytes. In XCOFF64, the string table contains the names of all symbols. If the string table is present, the first 4 bytes contain the length (in bytes) of the string table, including the length of this length field. The remainder of the table is a sequence of null-terminated ASCII strings. If the n_zeroes field in the symbol table entry is 0, then the n_offset field gives the byte offset into the string table of the name of the symbol.

If a string table is not used, it may be omitted entirely, or a string table consisting of only the length field (containing a value of 0 or 4) may be used. A value of 4 is preferable. The following table shows string table organization.
### Table 34. String Table Organization

<table>
<thead>
<tr>
<th>Offset</th>
<th>Length in Bytes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4</td>
<td>Length of string table.</td>
</tr>
<tr>
<td>4</td>
<td>n</td>
<td>Symbol name string, null-terminated.</td>
</tr>
</tbody>
</table>

Field repeats for each symbol name.

For general information on the XCOFF file format, see "XCOFF Object File Format."

### dbx Stabstrings

The debug section contains the symbolic debugger stabstrings (symbol table strings). It is generated by the compilers and assemblers. It provides symbol attribute information for use by the symbolic debugger.

See "Debug Section" for a general discussion.

### Stabstring Terminal Symbols

In the stabstring grammar, there are five types of terminal symbols, which are written in all capital letters. These symbols are described by the regular expressions in the following list:

**Note:** The [ ] (brackets) denote one instance, [ ]* (brackets asterisk) denote zero or more instances, [ ]+ (brackets plus sign) denote one or more instances, ( ) (parentheses) denote zero or one instance, .* (dot asterisk) denotes a sequence of zero or more bytes, and | (pipe) denotes alternatives.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Regular Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>[^:';&quot;] (A name consists of any non-empty set of characters, excluding ; : ’ or “.)</td>
</tr>
<tr>
<td>STRING</td>
<td>&quot;.&quot;</td>
</tr>
</tbody>
</table>

Within a string, the \ (backslash character) may have a special meaning. If the character following the \ is another \, one of the backslashes is ignored. If the next character is the quote character used for the current string, the string is interpreted as containing an embedded quote. Otherwise, the \ is interpreted literally. However, if the closing quote is the last character in the stabstring, and a \ occurs immediately before the quote, the \ is interpreted literally. This use is not recommended.

The \ must be quoted only in the following instances:
- The \ is the last character in the string (to avoid having the closing quote escaped).
- The \ is followed by the current quote character.
- The \ is followed by another \.

An escaped quote is required only when a single string contains both a single quote and a double quote. Otherwise, the string should be quoted with the quote character not contained in the strings.

A string can contain embedded null characters, so utilities that process stabstrings must use the length field to determine the length of a stabstring.

| INTEGER | (-)[0-9]+ |
| HEXINTEGER | [0-9A-F]+ |

The hexadecimal digits A-F must be uppercase.

| REAL | [HIDDD]/([+-][0-9]+)([+-][0-9]+)("[eEqQ]/([+-][0-9]+) | (+-))INF | QNAN | SNAN |

### Stabstring Grammar

REALs may be preceded by white space, and STRINGS may contain any characters, including null and blank characters. Otherwise, there are no null or blank characters in a stabstring.
Long stabstrings can be split across multiple symbol table entries for easier handling. In the stabstring grammar, a # (pound sign) indicates a point at which a stabstring may be continued. A continuation is indicated by using either the ? (question mark) or \ as the last character in the string. The next part of the stabstring is in the name of the next symbol table entry. If an alternative for a production is empty, the grammar shows the keyword /*EMPTY*/.

The following list contains the stabstring grammar:

**Stabstring:**

Basic structure of stabstring:

- **NAME** : **Class**
  - Name of object followed by object classification
  - **Class** Unnamed object classification.

**Class:** Object classifications:

- c = **Constant** ;
  - Constant object

**NamedType**

User-defined types and tags

**Parameter**

Argument to subprogram

**Procedure**

Subprogram declaration

**Variable**

Variable in program

- **Label** Label object.

**Constant:**

Constant declarations:

- b **OrdValue**
  - Boolean constant

- c **OrdValue**
  - Character constant

- e **TypeID**, **OrdValue**
  - Enumeration constant

- i **INTEGER**
  - Integer constant

- r **REAL**
  - Decimal or binary floating point constant

- s **STRING**
  - String constant

**C REAL, REAL**

Complex constant

- S **TypeID**, **NumElements**, **NumBits**, **BitPattern**
  - Set constant.

**OrdValue:**

Associated numeric value: **INTEGER**
**NumElements:**
Number of elements in the set: INTEGER

**NumBits:**
Number of bits in item: INTEGER

**NumBytes:**
Number of bytes in item: INTEGER

**BitPattern:**
Hexadecimal representation, up to 32 bytes: HEXINTEGER

**NamedType:**
User-defined types and tags:
- t *TypeId*
  User-defined type (TYPE or typedef), excluding those that are valid for T *TypeId*
- T *TypeId*
  Struct, union, class, or enumeration tag

**Parameter:**
Argument to procedure or function:
- a *TypeId*
  Passed by reference in general register
- p *TypeId*
  Passed by value on stack
- v *TypeId*
  Passed by reference on stack
- C *TypeId*
  Constant passed by value on stack
- D *TypeId*
  Passed by value in floating point register
- R *TypeId*
  Passed by value in general register
- X *TypeId*
  Passed by value in vector register

**Procedure:**
Procedure or function declaration:
- Proc *Proc*
  Procedure at current scoping level
- Proc , NAME : NAME
  Procedure named 1st NAME, local to 2nd NAME, where 2nd NAME is different from the current scope.

**Variable:**
Variable in program:
- *TypeId*
  Local (automatic) variable of type *TypeId*
- d *TypeId*
  Floating register variable of type *TypeId*
- h *TypeId*
  Static thread-local variable of type *TypeId*
r **TypeID**
    Register variable of type **TypeID**

x **TypeID**
    Vector register variable of type **TypeID**

G **TypeID**
    Global (external) variable of type **TypeID**

H **TypeID**
    Global (external) thread-local variable of type **TypeID**

S **TypeID**
    Module variable of type **TypeID** (C static global)

V **TypeID**
    Own variable of type **TypeID** (C static local)

Y FORTRAN pointer variable

Z **TypeID** **NAME**
    FORTRAN pointee variable

**Label:** Label:
L Label name.

**Proc:** Different types of functions and procedures:

f **TypeID**
    Private function of type **TypeID**

g **TypeID**
    Generic function (FORTRAN)

m **TypeID**
    Module (Modula-2, ext. Pascal)

J **TypeID**
    Internal function of type **TypeID**

F **TypeID**
    External function of type **TypeID**

I (capital i) Internal procedure

P External procedure

Q Private procedure

**TypeID:**
    Type declarations and identifiers:

    **INTEGER**
    Type number of previously defined type

    INTEGER = **TypeDef**
    New type number described by **TypeDef**

    INTEGER = **TypeAttrs** **TypeDef**
    New type with special type attributes

**TypeAttrs:**
    @ **TypeAttrList** ;

**Note:** Type attributes (**TypeAttrs**) are extra information associated with a type, such as alignment constraints or pointer-checking semantics. The **dbx** program recognizes only the **size**
attribute and the packed attribute. The size attribute denotes the total size of a padded element within an array. The packed attribute indicates that a type is a packed type. Any other attributes are ignored by dbx.

TypeDef:
Basic descriptions of objects:

INTEGER
Type number of a previously defined type

b TypeID ; # NumBytes
Pascal space type

c TypeID ; # NumBits
Complex type TypeID

d TypeID
File of type TypeID

e EnumSpec ;
Enumerated type (default size, 32 bits)

g TypeID ; # NumBits
Floating-point type of size NumBits

D TypeID ; # NumBits
Decimal floating-point type of size NumBits

For i types, ModuleName refers to the Modula-2 module from which it is imported.

i NAME : NAME ;
Imported type ModuleName:Name

k TypeID
C++ constant type

l ; # Usage-is-index; specific to COBOL

m OptVBaseSpec OptMultiBaseSpec TypeID : TypeID : TypeID ;
C++ pointer to member type; the first TypeID is the member type; the second is the type of the class
n  *TypeID* ; # *NumBytes*
   String type, with maximum string length indicated by *NumBytes*

o  NAME ;
   Opaque type

o  NAME , *TypeID*
   Opaque type with definition of *TypeID*

w  *TypeID*
   Wide character

z  *TypeID* ; # *NumBytes*
   Pascal gstring type

C  Usage
   COBOL Picture

I  *NumBytes* ; # *PicSize*
   (uppercase i) Index is type; specific to COBOL

K  *CobolFileDesc* ;
   COBOL File Descriptor

M  *TypeID* ; # *Bound*
   Multiple instance type of *TypeID* with length indicated by *Bound*

N  Pascal *Stringptr*

S  *TypeID*
   Set of type *TypeID*

*  *TypeID*
   Pointer of type *TypeID*

&  *TypeID*
   C++ reference type

V  *TypeID*
   C++ volatile type

Z  C++ ellipses parameter type

Array  Subrange  ProcedureType
   For function types rather than declarations

Record
   Record, structure, union, or group types

EnumSpec:
   List of enumerated scalars:

   EnumList
   Enumerated type (C and other languages)

   *TypeID* : EnumList
   C++ enumerated type with repeating integer type

EnumList:

Enum  EnumList Enum

Enum:  Enumerated scalar description:
   NAME : OrdValue , #

Array:  Array descriptions:
TypeID ; # TypeID
   Array; FirstTypeID is the index type

A TypeID
   Open array of TypeID

D INTEGER , TypeID
   N-dimensional dynamic array of TypeID

E INTEGER , TypeID
   N-dimensional dynamic subarray of TypeID

O INTEGER , TypeID
   New open array

P TypeID ; # TypeID
   Packed array

Subrange:
   Subrange descriptions:
   r TypeID ; # Bound ; # Bound
       Subrange type (for example, char, int,\,), lower and upper bounds

Bound:
   Upper and lower bound descriptions:

   INTEGER
       Constant bound

   Boundtype INTEGER
       Variable or dynamic bound; value is address of or offset to bound

   J
       Bound is indeterminable (no bounds)

Boundtype:
   Adjustable subrange descriptions:
   A    Bound passed by reference on stack
   S    Bound passed by value in static storage
   T    Bound passed by value on stack
   a    Bound passed by reference in register
   t    Bound passed by value in register

ProcedureType:
   Function variables (1st type C only; others Modula-2 & Pascal)
   f TypeID ;
       Function returning type TypeID
   f TypeID , NumParams ; TParamList ;
       Function of N parameters returning type TypeID
   p NumParams ; TParamList ;
       Procedure of N parameters
   R NumParams ; NamedTParamList
       Pascal subroutine parameter
   F TypeID , NumParams ; NamedTParamList ;
       Pascal function parameter

NumParams:
   Number of parameters in routine:
INTEGER.

TParamList:
   Types of parameters in Modula-2 function variable:
   TParam
      Type of parameter and passing method

TParam:
   Type and passing method
   TypeID , PassBy ; #

NamedTParamList:
   Types of parameters in Pascal-routine variable:
      /*EMPTY*/
NamedTPList

NamedTPList:
   NamedTParam NamedTPList NamedTParam

NamedTParam:
   Named type and passing method:
   Name : TypeID , PassBy InitBody ; #
   : TypeID , PassBy InitBody ; #
   Unnamed parameter

Record:
   Types of structure declarations:
      • s NumBytes # FieldList ;
      • Structure or record definition
      • u NumBytes # FieldList ;
      • Union
      • v NumBytes # FieldList VariantPart ;
      • Variant Record
      • Y NumBytes ClassKey OptPBV OptBaseSpecList ( ExtendedFieldListOptNameResolutionList ;
      • C++ class
      • G Redefinition , n NumBits # FieldList ;
      • COBOL group without conditionals

Gn NumBits FieldList ;
      • G Redefinition , c NumBits # CondFieldList ;
      • COBOL group with conditionals

Gc NumBits CondFieldList ;

OptVBaseSpec:
      v      ptr-to-mem class has virtual bases.
      /*EMPTY*/
      Class has no virtual bases.

OptMultiBaseSpec:
      m      Class is multi-based.
      /*EMPTY*/
      Class is not multi-based.

OptPBV:
 Class is always passed by value.
;/*EMPTY*/
    Class is never passed by value.

ClassKey:
  s  struct
  u  union
  c  class

OptBaseSpecList:
  /*EMPTY*/ BaseSpecList

BaseSpecList:
  BaseSpec
    BaseSpecList , BaseSpec

BaseSpec:
    VirtualAccessSpec BaseClassOffset : ClassTypeID

BaseClassOffset:
    INTEGER
        Base record offset in bytes

ClassTypeID:
   TypeID
        Base class type identifier

VirtualAccessSpec:
    v  AccessSpec
        Virtual
    v  Virtual
    AccessSpec
    /*EMPTY*/

GenSpec:
    c  Compiler-generated
    /*EMPTY*/

AccessSpec:
    i #  Private
    o #  Protected
    u #  Public

AnonSpec:
    a  Anonymous union member
    /*EMPTY*/

VirtualSpec:
    v p  Pure virtual
    v  Virtual
    /*EMPTY*/
ExtendedFieldList:
  ExtendedFieldList ExtendedField
  /*EMPTY*/

ExtendedField:
  GenSpec AccessSpec AnonSpec DataMember
  GenSpec VirtualSpec AccessSpec OptVirtualFuncIndex MemberFunction
  AccessSpec AnonSpec NestedClass
  AnonSpec FriendClass
  AnonSpec FriendFunction

DataMember:
  MemberAttrs : Field ;

MemberAttrs:
  IsStatic IsVtblPtr IsVBasePtr

IsStatic:
  /*EMPTY*/
  s  Member is static.

IsVtblPtr:
  /*EMPTY*/
  p INTEGER NAME
     Member is vtbl pointer; NAME is the external name of v-table.

IsVBasePtr:
  /*EMPTY*/
  b  Member is vbase pointer.
  r  Member is vbase self-pointer.

Member Function:
  [ FuncType MemberFuncAttrs : NAME :TypeID ; #

MemberFuncAttrs:
  IsStatic IsInline IsConst IsVolatile

IsInline:
  /*EMPTY*/
  i  Inline function

IsConst:
  /*EMPTY*/
  k  const member function

IsVolatile:
  /*EMPTY*/
  V  Volatile member function

NestedClass:
  N TypeID ; #

FriendClass:
  ( TypeID ; #

FriendFunction:
  ] NAME :TypeID ; #

OptVirtualFuncIndex:
  /*EMPTY*/ INTEGER
FuncType:
   f  Member function
   c  Constructor
   d  Destructor

InitBody:
   STRING
   /*EMPTY*/

OptNameResolutionList:
   /*EMPTY*/
   ) NameResolutionList

NameResolutionList: NameResolution
   NameResolution , NameResolutionList

NameResolution: MemberName : ClassTypeID
   Name is resolved by compiler.
   MemberName:
      Name is ambiguous.

MemberName:
   NAME

FieldList:
   Structure content descriptions:
   Field  /*EMPTY*/
   FieldList Field
      Member of record or union.

Field:
   Structure-member type description:
      NAME :TypeID , BitOffset , NumBits ; #

VariantPart:
   Variant portion of variant record:
      [ Vtag VFieldList ]
         Variant description

VTag:
   Variant record tag:
   ( Field Member of variant record
      ( NAME : ; #
         Variant key name

VFieldList:
   Variant record content descriptions:
      VList
      VFieldList VList
         Member of variant record
      VList: Variant record fields:
         VField
         VField VariantPart
            Member of variant record

VField:
   Variant record member type description:
( VRangeList : FieldList
    Variant with field list

VRangeList:
    List of variant field labels:
    VRange
    VRangeList , VRange
    Member of variant record

VRange:
    Variant field descriptions:
    b OrdValue
        Boolean variant
    c OrdValue
        Character variant
    e TypeID , OrdValue
        Enumeration variant
    i INTEGER
        Integer variant
    r TypeID ; Bound ; Bound
        Subrange variant

CondFieldList:
    Conditions,#FieldList
    FieldList# ;

Conditions:
    /*Empty*/
    Conditions condition

BitOffset:
    Offset in bits from beginning of structure: INTEGER

Usage:
    Cobol usage description:
    PICStorageType NumBits , EditDescription , PicSize ;
    Redefinition , PICStorageType NumBits , EditDescription , PicSize ;
    PICStorageType NumBits , EditDescription , PicSize , # Condition ;
    Redefinition , PICStorageType NumBits , EditDescription , PicSize , # Condition ;

Redefinition:
    Cobol redefinition: r NAME

PICStorageType:
    Cobol PICTURE types:
    a Alphabetic
    b Alphabetic, edited
    c Alphanumeric
    d Alphanumeric, edited
    e Numeric, signed, trailing, included
    f Numeric, signed, trailing, separate
    g Numeric, signed, leading, included
    h Numeric, signed, leading, separate
i Numeric, signed, default, comp
j Numeric, unsigned, default, comp
k Numeric, packed, decimal, signed
l Numeric, packed, decimal, unsigned
m Numeric, unsigned, comp-x
n Numeric, unsigned, comp-5
o Numeric, signed, comp-5
p Numeric, edited
q Numeric, unsigned
s Indexed item
t Pointer

_EditDescription:_
Cobol edit description:

STRING
Edit characters in an alpha PIC

INTEGER
Decimal point position in a numeric PIC

_PicSize:_
Cobol description length:

INTEGER
Number of repeated '9's in numeric clause, or length of edit format for edited numeric

_Condition:_
Conditional variable descriptions:

NAME : INTEGER = q ConditionType , ValueList ; #

ConditionType:
Condition descriptions:

ConditionPrimitive , KanjiChar

ConditionPrimitive:
Primitive type of Condition:

n Sign DecimalSite
   Numeric conditional

   a Alphanumeric conditional

   f Figurative conditional

Sign: For types with explicit sign:

   + Positive

   - Negative

   [^+-] Not specified

DecimalSite:
   Number of places from left for implied decimal point:

   INTEGER
KanjiChar:
  0 only if Kanji character in value: INTEGER

ValueList
  Values associated with condition names

Value ValueList Value
Value Values associated with condition names:
  INTEGER : ArbitraryCharacters #
  Integer indicates length of string

CobolFileDesc:
  COBOL file description:
  Organization AccessMethod NumBytes

Organization:
  COBOL file-description organization:
    i Indexed
    l Line Sequential
    r Relative
    s Sequential

AccessMethod:
  COBOL file description access method:
    d Dynamic
    o Sort
    r Random
    s Sequential

PassBy:
  Parameter passing method:
    INTEGER
    0 = passed-by reference; 1 = passed-by value

Related Information

Header Files.
The as command, dbx command, dump command, ld command, size command, strip command, and what command.
Chapter 3. Special Files

A special file is associated with a particular hardware device or other resource of the computer system. The operating system uses special files, sometimes called device files, to provide file I/O access to specific character and block device drivers.

Special files, at first glance, appear to be just like ordinary files, in that they:
• Have path names that appear in a directory.
• Have the same access protection as ordinary files.
• Can be used in almost every way that ordinary files can be used.

However, there is an important difference between the two. An ordinary file is a logical grouping of data recorded on disk. A special file, on the other hand, corresponds to a device entity. Examples are:
• An actual device, such as a line printer
• A logical subdevice, such as a large section of the disk drive
• A pseudo device, such as the physical memory of the computer (/dev/mem) or the null file (/dev/null).

Special files are distinguished from other files by having a file type (c or b, for character or block) stored in the i-nodes to indicate the type of device access provided. The i-node for the special file also contains the device major and minor numbers assigned to the device at device configuration time.

Note: Data corruption, loss of data, or loss of system integrity (a system crash) will occur if devices supporting paging, logical volumes, or mounted file systems are accessed using block special files. Block special files are provided for logical volumes and disk devices on the operating system and are solely for system use in managing file systems, paging devices, and logical volumes. These files should not be used for other purposes.

Several special files are provided with the operating system. By convention, special files are located in the /dev directory.

More information is provided about the following special files:

<table>
<thead>
<tr>
<th>Special File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3270cn</td>
<td>Provides access to 3270 connection adapters by way of the 3270 connection adapter device driver.</td>
</tr>
<tr>
<td>bus</td>
<td>Provides access to the hardware bus by way of the machine I/O device driver.</td>
</tr>
<tr>
<td>cd</td>
<td>Provides access to the cdrom device driver.</td>
</tr>
<tr>
<td>console</td>
<td>Provides access to the system console.</td>
</tr>
<tr>
<td>dials</td>
<td>Provides access to the dials.</td>
</tr>
<tr>
<td>dump</td>
<td>Supports system dump.</td>
</tr>
<tr>
<td>entn</td>
<td>Provides access to the 3COM Ethernet adapters by way of the Ethernet device handler for this platform.</td>
</tr>
<tr>
<td>error</td>
<td>Supports error logging.</td>
</tr>
<tr>
<td>fd</td>
<td>Provides access to the diskette device driver.</td>
</tr>
<tr>
<td>fddin</td>
<td>Provides access to the FDDI device driver by way of the FDDI device handler.</td>
</tr>
<tr>
<td>GIO</td>
<td>Provides access to the graphics I/O (GIO) adapter.</td>
</tr>
<tr>
<td>ide</td>
<td>Provides access to the Integrated Device Electronics (IDE) adapter driver.</td>
</tr>
<tr>
<td>kbd</td>
<td>Provides access to the natively attached keyboard.</td>
</tr>
<tr>
<td>kmem and mem</td>
<td>Provides privileged read and write access to virtual memory.</td>
</tr>
<tr>
<td>lft</td>
<td>Implements a low-function terminal (LFT) device.</td>
</tr>
<tr>
<td>ide</td>
<td>Provides access to the IDE adapter device driver.</td>
</tr>
<tr>
<td>lp</td>
<td>Provides access to the line printer device driver.</td>
</tr>
<tr>
<td>lpfk</td>
<td>Provides access to the lighted program function key (LPFK) array.</td>
</tr>
<tr>
<td>lvdd</td>
<td>Provides access to the logical volume device driver.</td>
</tr>
</tbody>
</table>
**Related Information**

[File Formats Overview](#) defines and describes file formats.

[Header Files Overview](#) describes header files.

### 3270cn Special File

**Purpose**

Provides access to 3270 connection adapters by way of the 3270 connection adapter device handler.

**Description**

The 3270cn character special file provides access to the 3270 connection adapter device handler for the purpose of emulating 3270 display stations and printers. The device handler is a multiplexed device handler that supports an independent logical 3270 session on each of its channels.

The device handler supports two modes of operation:

**Distributed Function Terminal (DFT) mode**

In DFT mode, the adapter can appear as multiple SNA or non-SNA display sessions, non-SNA printer sessions, or both, and is an intelligent device to the control unit. In this mode, the device handler provides the capability of emulating several 3278/79 display stations. If the attached control unit does not support Extended Asynchronous Event Device Status, either the control unit port or the device handler must be configured for one session only.

**3278/79 emulation Control Unit Terminal (CUT) mode**

In CUT mode, the adapter appears as a single-session, unintelligent device to the control unit. In this mode, the device handler provides the capability of emulating a single 3278/79 display station.
The device handler supports up to four 3270 connection adapters, each of which may have up to five DFT sessions or one CUT session.

The /usr/include/sys/io3270.h file contains the definitions of the structures used by the device handler.

Usage Considerations
When accessing the 3270 connection device handler, the following should be taken into account:

**Driver initialization and termination**

The device handler may be loaded and unloaded. The device handler supports the configuration calls to initialize and terminate itself, but does not support the configuration call to query vital product data (VPD). Subroutines other than open and close are discussed in regard to the mode in which the device handler is operating.

**Special file support**

Subroutines other than open and close are discussed in regard to the mode in which the device handler is operating.

**Subroutine Support**

The 3270 device handler provides 3270-specific support for the following subroutines:

- open
- close
- read
- readx (non-SNA DFT mode only)
- write
- writex (non-SNA DFT mode only)
- ioctl

**open and close Subroutines:** The device handler supports the 3270cn special file as a character-multiplex special file. The special file must be opened for both reading and writing (O_RDWR).

A special consideration exists for closing the 3270cn special file. If the file was opened in both CUT mode and CUT-File Transfer mode, the close operation for CUT-File Transfer mode must precede the close operation for CUT mode.

The special file name used in an open call takes on several different forms, depending on how the device is to be opened. Types of special file names are:

- **dev/3270cn/C**
  - Starts the device handler in CUT mode for the selected port, where the value of \( n \) is \( 0 \leq n \leq 7 \).

- **/dev/3270cn/F**
  - Starts the device handler in CUT File-Transfer mode for the selected port, where the value of \( n \) is \( 0 \leq n \leq 7 \). The file must be currently open in CUT mode before it can be opened in CUT File-Transfer mode.

- **/dev/3270cn/***
  - Starts the device handler in DFT mode for the selected port, where the value of \( n \) is \( 0 \leq n \leq 7 \) and the * (asterisk) is defined by \( P/a \), as follows:

    - \( P/00 \), \( P/01 \), \( P/02 \), ..., \( P/1F \)
      - The printer session specified by the \( P \) variable is equal to the control unit session address, and the value of \( a \) is less than or equal to \( 0x1F \).

      - **01 through 05**
        - Terminal session number.

- **/dev/3270cn**
  - Starts the device handler in DFT mode for the selected port, where the value of \( n \) is \( 0 \leq n \leq 7 \).

**read Subroutine in Non-SNA DFT Mode:** Data received by the communication adapter from the host is placed in the buffer until the message is completed or the buffer is full. When either condition occurs, the
driver returns program control back to the application. The application can determine the status of a read subroutine call by issuing a WDC_INQ ioctl operation.

If the WDC_INQ operation returns a status indicating that more data is available, the application should immediately issue another read call. Available data must be read as soon as possible to avoid degrading link or host performance.

If a read call is made and no data is available, the calling process is blocked until data becomes available. To avoid blocking, use the poll subroutine to determine if data is available.

The host sends data as an outbound 3270 data stream. The device handler translates the command codes in the outbound 3270 data stream. The command codes and translations are as follows:

<table>
<thead>
<tr>
<th>Command Code</th>
<th>Into Driver</th>
<th>Out of Driver</th>
</tr>
</thead>
<tbody>
<tr>
<td>Erase All Unprotected</td>
<td>0x6F</td>
<td>0x0F</td>
</tr>
<tr>
<td>Erase/Write</td>
<td>0xF5</td>
<td>0x03</td>
</tr>
<tr>
<td>Erase/Write Alternate</td>
<td>0x7E</td>
<td>0x0D</td>
</tr>
<tr>
<td>Read Buffer</td>
<td>0xF2</td>
<td>0x02</td>
</tr>
<tr>
<td>Read Modified</td>
<td>0xF6</td>
<td>0x06</td>
</tr>
<tr>
<td>Write</td>
<td>0xF1</td>
<td>0x01</td>
</tr>
<tr>
<td>Write Structured Field</td>
<td>0xF3</td>
<td>0x11</td>
</tr>
</tbody>
</table>

**read Subroutine in SNA DFT Mode:** The communication adapter receives data from the control unit in individual SNA data segments. The device driver notifies the application that data is available. During the read subroutine call, the data is transferred to the application’s user space from the device driver’s kernel space (without the TCA header from the control unit), and control is passed back to the application. The device driver acknowledges each SNA data segment received, making it unnecessary for the application to inquire about the link status after the read call.

**Note:** The STAT_ACK ioctl operation is not valid in SNA DFT mode.

Unlike non-SNA DFT mode, neither chaining nor command interpretation is performed by the device driver in SNA DFT mode. The application must both accumulate SNA data segments to form an response unit (RU) and interpret any 3270 data contained within.

**readx Subroutine in Non-SNA DFT Mode:** Data received by the communication adapter from the host is placed in the buffer until either the message completes or the buffer is full. Upon completion of the read call, the io3270 structure pointed to by the read extension contains the status. One of the following status codes is set in the io_flags field of the io3270 structure:

- **WDI_DAVAIL**: Additional data is available for this link address.
- **WDI_COMM**: A communication error occurred. The io_status field contains the corresponding message code.
- **WDI_PROG**: A program error occurred. The io_status field contains the corresponding message code.
- **WDI_MACH**: A hardware error occurred. The io_status field contains the corresponding message code.
- **WDI_FATAL**: An error occurred that prevents further communication with the host. This flag is optionally set in addition to the WDI_COMM, WDI_PROG, or WDI_MACH flag. It is also set when a coax failure occurs. In this case, the io_status field contains a value of WEB_610, but the WDI_COMM, WDI_PROG, or WDI_MACH flag is not set.

When reset, the WDI_DAVAIL flag indicates that the data just read marks the completion of an outbound 3270 data stream.
If the WDI_DAVAIL flag indicates more data is available, another readx subroutine should be issued immediately. Available data must be read as soon as possible to avoid degrading link or host performance.

If a readx subroutine call is made and no data is available, the calling process is blocked until data becomes available. To avoid blocking, use the poll subroutine to determine if data is available.

Data received from the host is in the form of an outbound 3270 data stream. The device driver translates the command codes in the outbound 3270 data stream.

**Note:** The 3270 write commands require the application to send a status to the host. Status is sent using the WDC_SSTAT ioctl operation.

**write Subroutine in Non-SNA DFT Mode:** In non-SNA DFT mode, the write subroutine sends an inbound 3270 data stream to the host. The buffer specified on a write subroutine call must contain a complete inbound 3270 data stream. The write call is complete when it has successfully transferred from the buffer specified on the subroutine call.

**write Subroutine in SNA DFT Mode:** In SNA DFT mode, the write subroutine transmits SNA data to the host system. This data can be either a 3270 data stream with SNA headers or an SNA response.

The application sends data to the device driver, one RU at a time. The device driver is then responsible for segmenting the inbound SNA data. If a second write call is made before the first call is processed, the second call does not proceed until the device driver is ready. After the data is transferred from the application’s user space to the device driver’s kernel space, the write subroutine completes and control is returned to the application.

If the device driver detects a coax disconnect between two write calls, the second write call will return to the application, with the errno global variable set toEFAULT.

**writex Subroutine in Non-SNA DFT Mode:** The writex subroutine sends an inbound 3270 data stream to the host. The buffer specified on a writex subroutine call must contain a complete inbound 3270 data stream.

The write subroutine is complete when it has successfully transferred the data from the specified buffer. Upon completion of the write subroutine call, the io3270 structure pointed to by the write extension contains the status. One of the following status codes is set in the io_flags field of the io3270 structure:

- **WDI_DAVAIL** Indicates that data is available for this link address; the data must be read before any write can occur.
- **WDI_COMM** Indicates a communication error. The io_status field contains the corresponding message code.
- **WDI_PROG** Indicates a program error. The io_status field contains the corresponding message code.
- **WDI_MACH** Indicates a hardware error. The io_status field contains the corresponding message code.

**ioctl Subroutine in DFT Mode:** The ioctl subroutine may be issued to the device handler when it is in DFT mode. The following are the available ioctl operations:

- **IOCINFO** Returns the logical terminal number. This number is the EBCDIC representation of the controller type and the controller attachment protocol in the iocinfo structure.
- **WDC_AUTO** Valid only for non-SNA DFT mode. Provides the handler with the option to automatically acknowledge the receipt of a valid 3270 data stream. An acknowledgment is sent only if the beginning of the 3270 data stream consists of 0xF3 00 06 40 00 F1 C2 xx xx 10 14, where the xx fields are not examined. This command also allows the driver not to indicate acknowledgment upon receipt of data.
WDC_INQ  Queries the status of the last non-SNA read or write call issued by the application. Also, the WDC_INQ operation determines if data is available for reading. The status is placed in the io_flags field of the io3270 structure. This field accepts the following values:

**WDI_DAVAIL**
- Data is available for reading. The data is buffered either in the driver or in the communication adapter. The data should be read immediately to avoid its having an impact on performance.

  In non-SNA DFT mode, a write or writex subroutine call cannot complete until the data has been read. In SNA DFT mode, the WDI_DAVAIL flag is used only to indicate that data is available when the device driver wakes up the application (if waiting on a poll or select call) after receiving data from the control unit.

**WDI_COMM, WDI_PROG, or WDI_MACH**
- Indicates a communication check, program check, or machine check, respectively. In each of these cases, the io_status field contains a message code that specifies the type of check.

**WDI_FATAL**
- Indicates that an error has occurred that prevents further communication between the application and the device driver, typically a coax disconnect or adapter failure. This flag may be set in conjunction with the WDI_COMM, WDI_PROG, or WDI_MACH flag. If the communications failure was caused by a coax disconnect, the io_status field contains a value of WEB_610.

**WDI_WCUS_30**
- A communications check reminder that occurs when there is a network failure and the control unit is still communicating with the communication adapter. The specific type of error is contained in the io_status field as a 5XX error code. The communications check reminder is cleared automatically after the network condition is corrected.

**WDI_WCUS_31**
- Indicates that the communications check reminder has been cleared.

**WDI_CU**
- Valid only for SNA DFT mode. Indicates that an ACTLU or DACTLU request was received by the device driver. The accompanying data is contained in the io_extra field of the io3270 structure.

**WDC_POR**
- The link address is first disabled and then re-enabled to emulate a 3270 terminal power-on reset function.

**WDC_SSTAT**
- Valid only for non-SNA DFT mode. Sends status to the host. The argument field contains one of the following values:

  **STAT_ACK**
  - The previously received 3270 data stream is valid, and the proper response is made to the host.

  **STAT_RESET**
  - Sends a RESET Key to the DFT device handler.

  **STAT_PRTCMP**
  - The printer session has completed printing the data.

  **STAT_BERR**
  - Received a bad buffer order or an invalid buffer address.

  **STAT_UNSUP**
  - Received an unsupported 3270 command.

  The /usr/include/sys/io3270.h file contains the definitions of the structures used by the device handler.

**Error Conditions in DFT Mode:** The following error conditions may be returned when accessing the device handler through the 3270cn special file:
EBUSY  An open was requested for a channel that is already open.
EFAULT  A buffer specified by the caller was not valid.
EINTR  A subroutine call was interrupted.
EINVAL  An invalid argument was received.
EIO  An unrecoverable I/O error occurred on the requested data transfer.
ENODEV  An open was requested for an invalid channel.
ENOMEM  The driver could not allocate memory for use in the data transfer.
ENXIO  An operation was requested for an invalid minor device number.

**read Subroutine in CUT Mode:** The *read* subroutine places data received by the communication adapter in a buffer.

**Note:** To set the offset into the communication adapter's buffer from which to read, use the **EMSEEK** ioctl operation.

Two ioctl operations control the way the *read* subroutine operates: the **EMNWAIT** and **EMWAIT** operations. The **EMNWAIT** operation indicates that subsequent read calls should be satisfied immediately. The **EMWAIT** ioctl operation (the default) indicates that read calls should be satisfied only after an interrupt from the control unit indicates that something has changed on the display. The following are control unit interrupts:

- **Buffer Modification Complete**
- **Load I/O Address Command Decoded**

The *read* subroutine returns the number of bytes requested. The *read* subroutine returns 0 for the number of bytes read.

**write Subroutine in CUT Mode:** The *write* subroutine sends an inbound 3270 data stream to the host. The buffer specified on a *write* subroutine must contain a complete inbound 3270 data stream. To set the offset into the communication adapter buffer to begin to write, use the **EMSEEK** ioctl operation.

**ioctl Subroutine in CUT Mode:** The *ioctl* subroutine may be issued to the device handler in CUT mode. The following are acceptable *ioctl* operations:

- **EMKEY** Sends a scancode to the emulation adapter. The scan code is logically ORed with the **EMKEY** operation, and the result is used as the command field on the *ioctl* subroutine call.
- **EMCPOS** Returns the position of the cursor relative to the start of the communication adapter buffer.
- **EMXPOS** Disables the link address and then re-enables it to emulate a 3270 terminal power-on reset function.
- **EMNWAIT** Specifies that *read* subroutine calls should be satisfied immediately.
- **EMWAIT** Specifies that *read* subroutine calls should be satisfied only after a change to the emulation buffer or the cursor position (this is the default setting).
- **EMVISND** Returns the current contents of the emulation Visual/Sound register in the integer field. The address of this field is specified as the argument to the **EMVISND** operation.
- **EMIMASK** Provides a mask to specify which interrupts appear. The argument field specifies the address of the mask. The low-order bits of the mask (0 through 7) correspond to bits 0 through 7 of the Interrupt Status register. Bits 8 through 15 of the mask correspond to bits 0 through 7 of the Visual/Sound register.

This operation allows the driver to ignore visual or sound interrupts except for those bits specifically masked ON. When a bit is on, the interrupt that corresponds to that bit position appears. Interrupts that correspond to off (0) bit positions in the mask are discarded by the device handler. The previous mask setting is returned to the caller in the mask field. The interrupt status bits and the visual or sound bits are documented in the *IBM 3270 Connection Technical Reference*.

- **IOCINFO** Returns a structure of device information, as defined in the `/usr/include/sys/devinfo.h` file, to the user-specified area. The devtype field has a value of **DD_EM78**, which is defined in the `devinfo.h` file, and the flag field value has a value of 0.
- **EMSEEK** Sets the offset into the communication adapter buffer to begin a *read* or *write* subroutine call.
**Error Conditions in CUT Mode:** The following error conditions may be returned when accessing the device handler through the `dev/3270cn` special file:

- **EBUSY**: An open was requested for a channel that is already open. The keystroke buffer is full.
- **EFAULT**: A buffer specified by the caller is not valid.
- **EINTR**: A subroutine call was interrupted.
- **EINVAL**: An invalid argument was specified on an `ioctl` call.
- **EL3RST**: A reset command was received by the communications adapter.
- **ENOCONNECT**: The connection to the control unit stopped while a `read` operation, for which the `EMWAIT` `ioctl` operation had been specified, was waiting.
- **EIO**: An unrecoverable I/O error occurred on the requested data transfer.
- **ENXIO**: An operation was requested for a minor device number that is not valid.

This special file requires the IBM 3270 Connection Adapter.

**Related Information**

Special Files Overview.

The `close` subroutine, `open` subroutine, `poll` subroutine, `read` subroutine, `write` subroutine, `ioctl` subroutine.

---

**bus Special File**

**Purpose**

Provides access to each of the hardware buses by way of the machine I/O device driver.

**Description**

The `bus` special files consist of a pseudo-driver in the kernel that allows a privileged user to access each hardware I/O bus. This is done indirectly by using the `ioctl` subroutine. The calling process, however, must have the appropriate system privilege to open the `bus` special files.

For additional information on `bus` special files, see device configuration documentation in *AIX 5L Version 5.3 Kernel Extensions and Device Support Programming Concepts* and machine device driver documentation in *AIX 5L Version 5.3 Technical Reference: Kernel and Subsystems Volume 1*.

This capability should be used only by device initialization and configuration programs. Programs that depend upon the `bus` device interface may not be portable to machines with different hardware.

There is at least one `bus` special file, usually the `/dev/pci0` or the `/dev/bus0` special file. This file accesses the primary hardware bus. One `bus` special file exists for each hardware bus on the machine. Each `bus` special file gains access to the corresponding hardware bus, and exists only if the hardware bus is present or was present at one time. Run the following command to generate a list of all the defined `bus` special files for a machine:

```
lsdev -C -c bus -F name | xargs -i echo /dev/{}/
```

**Related Information**

The `ioctl` subroutine.
cd Special File

Purpose
Provides access to the CD-ROM device driver.

Description
The CD-ROM special file provides block and character (raw) access to disks in the CD-ROM drives.

The r prefix on a special file name means the drive is accessed as a raw device rather than a block device. Performing raw I/O with a compact disk requires the performance of all data transfers in multiples of the compact-disk logical block length. Also, all [seek] subroutines made to the raw CD-ROM device driver must set the file offset pointer to a value that is a multiple of the specified logical block size.

CD-ROM Device Drivers
Compact disks, used in CD-ROM device drivers, are read-only media that provide storage for large amounts of data. The special files /dev/cd0, /dev/cd1,... provide block access to compact disks. The special files /dev/rcd0, /dev/rc1,... provide character access.

When a CD-ROM disc is ejected from the drive for a mounted CD-ROM file system, the files on the compact disc can no longer be accessed. Before these files can be accessed again, the file systems mounted from the CD-ROM must be unmounted. Processes having files open on these file systems should be exited. Processes having current directories on these file systems should be moved. If these actions do not work, perform a forced unmount.

Another problem that results from ejecting the CD-ROM disc for a mounted CD-ROM file system is that the man command can become unresponsive. Reinserting the CD-ROM disc will not fix the problem. All processes (graphical and ASCII) should be exited and the file system should be forced unmounted and mounted again. Afterwards, any man commands can be started again.

Device-Dependent Subroutines
Most CD-ROM operations are implemented using the open, read, and close subroutines. However, for some purposes, use of the openx (extended) subroutine is required.

openx Subroutine
The openx subroutine is supported to provide additional functions to the open sequence. The openx subroutine requires appropriate authority to start. Attempting to execute this subroutine without the proper authority results in a return value of -1, with the errno global variable set to EPERM.

ioctl Subroutine
The IOCINFO ioctl operation is defined for all device drivers that use the ioctl subroutine. The remaining ioctl operations are all physical volume device-specific operations. Diagnostic mode is not required for the following operation. The IOCINFO operation returns a devinfo structure, which is defined in the devinfo.h file.

Error Codes
In addition to the error codes listed for the ioctl, open, read, and write subroutines, the following error codes are also possible:

EACCES A subroutine other than ioctl or close was attempted while in Diagnostic mode.
EACCES A normal read call was attempted while in Diagnostic mode.
EFAULT Illegal user address.
EBUSY The target device is reserved by another initiator.
EINVAL The device was opened with a mode other than read-only.
EINVAL  An nbyte parameter to a read subroutine is not an even multiple of the block size.
EINVAL  A sense-data buffer length greater than 255 is not valid for a CDIOCMD ioctl operation.
EINVAL  A data buffer length greater than that allowed by the drive is not valid for a CDIOCMD ioctl operation.
EINVAL  An attempt was made to configure a device that is still open.
EINVAL  An illegal configuration command has been given.
EMFILE  An open call has been attempted for a SCSI adapter that already has the maximum permissible number of open devices.
ENOTREADY  There is no compact disk in the drive.
ENODEV  An attempt was made to access a device that is not defined.
ENODEV  An attempt was made to close a device that has not been defined.
EMEDIA  The media was changed.
EIO  Hardware error or aborted command or illegal request.
EIO  An attempt has been made to read beyond the end of media.
EPERM  This subroutine requires appropriate authority.
ESTALE  A CD-ROM disk was ejected (without first being closed by the user) and then either re-inserted or replaced with a second disk.
ETIMEDOUT  An I/O operation has exceeded the given timer value.

Related Information

The close subroutine, ioctl subroutine, lseek subroutine, open subroutine, read subroutine.


console Special File

Purpose

Provides access to the system console.

Description

The /dev/console special file provides access to the device or file designated as the system console. This file can be designated as the console device by the person administering the system or a user with the appropriate permissions. The console character special file provides access to the console device driver. The console device driver in turn directs input and output to the device or file selected as the system console.

The system console is typically a terminal or display located near the system unit. It has two functions in the operating system. First, it provides access to the system when it is operating in a non-multiuser mode. (This would be the case during maintenance and diagnostic sessions.) A console login is also normally provided on this device for all operating system run levels.

Second, the system console displays messages for system errors and other problems requiring intervention. These messages are generated by the operating system and its various subsystems when starting or operating. The system console can also be redirected to a file or to the /dev/null special file for systems operating without a console device.

Console Driver Configuration Support

Console driver configuration support allows the system console to be assigned or reassigned to a specified device or file. Such support also provides query functions to determine the current and
configured path names for the device or file designated as the console. This configuration support is used by the \texttt{swcons}, \texttt{chcons}, and \texttt{lscons} high-level system management commands. It is also used by the console configuration method at system startup.

The \texttt{swcons} (switch console) command can be used during system operation to switch the system console output to a different target temporarily. This command switches only system information, error, and intervention-required messages to the specified destination. The \texttt{swcons} command does not affect the operation of the system console device that provides a login through the \texttt{getty} command. The device or file specified when using the \texttt{swcons} command remains the target for console output until one of the following happens:

- Another \texttt{swcons} command is issued.
- The system is started again.
- The console driver detects an error when accessing the designated device or file.

If an open or write error is detected on the device or file specified by the \texttt{swcons} command, the console device driver switches all output back to the device or file providing console support when the system started.

The \texttt{chcons} (change console) command can be used to switch the system console output to a different device or file for the next startup. This command does not affect the current console selection, but becomes effective when the system is started again.

When requested to activate a login on the console device, the \texttt{getty} program (which provides login support) uses the console configuration support to determine the path name of the targeted console device used at startup. This action ensures that the \texttt{swcons} command does not affect the console device being used for login.

Usage Considerations

The \texttt{open}, \texttt{close}, \texttt{read}, \texttt{write}, \texttt{ioctl}, \texttt{select}, and \texttt{poll} subroutines are supported by the console device driver and may be used with the \texttt{/dev/console} special file. These subroutines are redirected to the device or file serving as the current system console device by the console device driver.

**open and close Subroutines:** When an \texttt{open} subroutine call is issued to the console device driver, it is redirected to the device or file currently chosen as the console device. If the system console choice is a file, the file is opened with the \texttt{append} and \texttt{create} options when the first open of the \texttt{dev/console} file is received. Subsequent opens have no effect when the console selection is a file. However, the opens are then passed to the device driver supporting the device chosen as the console.

If the console selection has been temporarily switched using the \texttt{swcons} command and the first open of the new underlying device fails, the console device driver switches back to the console device or file with which the system was booted. This prevents important system messages from being lost.

An \texttt{ext} parameter passed using the \texttt{openx} subroutine is passed to the device driver supporting the console target or else ignored. (The latter is true if the console selection is a file.)

The \texttt{close} subroutine support is standard.

**select, poll, and ioctl Subroutines:** The \texttt{select}, \texttt{poll}, and \texttt{ioctl} subroutines are redirected to the current system console device when the console selection is not a file. If the selected console device is a file, the console device driver returns an error indicating that the subroutine is not supported.

An \texttt{ext} parameter passed to the \texttt{ioctlx} subroutine is then passed to the device driver supporting the console target, or else ignored. (The latter is true if the console selection is a file.)

**read and write Subroutines:** The \texttt{write} subroutine calls are redirected to the current console device or file. If the console selection has been temporarily switched using the \texttt{swcons} command, and the write to
the targeted device or file is unsuccessful, the console device driver switches back to the console device
or file from which the system was started and tries the write again. This prevents important system
messages from being lost in case the temporary console target is unavailable or unsuccessful. The
console device driver should stay connected to the original system device until another swcons command
is issued.

If the current console selection is a device, it redirects the read subroutine call. If the current console
selection is a file, the read call is rejected with an error (EACCES).

An ext parameter passed to the readx or writex subroutine is passed to the device driver supporting the
console target, or else ignored. (The latter is true if the console selection is a file.)

**Console Output Logging**

All output sent to the console is logged to a system log file. Only output sent to the console is logged. Any
output sent to a device acting as the console is not logged. This means that system informational, error,
and intervention-required messages are captured (logged), while other types of output seen at the console
are not; e.g., getty output, smitty output, user interaction at the console device, etc.

The log file is based on the alog format; this format allows the file to wrap after it attains a predetermined
maximum size. The alog command is typically used to view the console log file. The console log file
deviates from the normal alog format in that each record of the file contains, in addition to the logged text,
the user id who wrote to the console and the epoch time when it was written. The epoch time is formatted
and displayed in the user’s locale date and time when the file is output by the alog command.

When the console device is configured or when any modification is made to the console log file, ownership
of the file is set to root and permissions are set to 622 to match that of the console device driver special
file. The root user can modify the ownership or permissions, but they will not persist across boots.

The swcons command is used to make changes to console logging parameters during system operation;
these changes are rescinded at the next console device configuration (typically reboot), and the original
console logging parameters are reinstated.

The chcons command is used to make changes to the console logging parameters for the next console
device configuration (typically reboot). These changes do not apply to the current running system.

The console logging facility can also be configured using the alog command. When the alog -C flag is
used, changes are effective in the current running system and are persistent across boots. When the -s
flag is used (without) the -C) to change the file size, the file is changed immediately but this change is not
saved in the ODM and is not persistent across boots.

The parameters that control the console logging facility are the pathname of the log file, the maximum size
of the log file, and the verbosity index for logging. Restrictions on these parameters are:

- the log file path must be absolute
- the maximum file size must not exceed the current free space of the file system on which it is stored
  (and the user entered value is rounded up to the nearest 4K boundary)
- verbosity values are 0-9 with any value greater than 0 indicating that all console output is to be
  recorded.

**Console Output Tagging**

A facility is provided to prefix each console output message with the effective user ID of the user that sent
the message to the console. Only output sent to the console is tagged, any output sent to the device
acting as the console is not.
Both the `swcons` command and the `chcons` commands can be used to enable and disable console output tagging with the same caveats about the persistence of the values applying as mentioned above in Console Output Logging.

The console output tagging verbosity value is limited to the range 0-9. Any value greater than 0 causes all console output to be tagged.

**Files**

`/dev/null` Provides access to the null device.

**Related Information**
The `chcons` command, `getty` command, `lscons` command, `swcons` command, `alog` command.

The `consdef` file.

The `close` subroutine, `ioctl` subroutine, `lseek` subroutine, `open` subroutine, `poll` subroutine, `read` subroutine, `select` subroutine, `write` subroutine.

---

**dials Special File**

**Purpose**
Provides access to the dials.

**Description**
The dials special file is the application interface to the dials. It allows the applications to receive operator input from the dials and to set the granularity of the dials.

**Configuration**
Standard configuration methods are provided for the dials special file. The user cannot enter configurable attributes by way of the command line.

**Usage Considerations**

**open:** An `open` subroutine call specifying the dials special file is processed normally except that the `Oflag` and `Mode` parameters are ignored. An open request is rejected if the special file is already opened or if a kernel extension attempts to open the dials special file. All dials inputs are flushed following an open call until an input ring is established.

**read and write:** The dials special file does not support read or write subroutine calls. Input data is obtained from the dials via the input ring. The read and write subroutine calls behave the same as read or write to `/dev/null`. See “LFT Input Ring” in *AIX 5L Version 5.3 Kernel Extensions and Device Support Programming Concepts* for how to use the input ring.

**ioctl:** The dials special file supports the following `ioctl` operations:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOCINFO</td>
<td>Returns the <code>devinfo</code> structure.</td>
</tr>
<tr>
<td>DIALREGRING</td>
<td>Registers input ring.</td>
</tr>
<tr>
<td>DIALRFLUSH</td>
<td>Flushes input ring.</td>
</tr>
<tr>
<td>DIALSETGRAND</td>
<td>Sets dial granularity.</td>
</tr>
</tbody>
</table>
Error Codes
The error codes can be found in the /usr/include/sys/errno.h file.

EFAULT Indicates insufficient authority to access address or invalid address.
EIO Indicates I/O error.
ENOMEM Indicates insufficient memory for required paging operation.
ENOSPC Indicates insufficient file system or paging space.
EINVAL Indicates invalid argument specified.
EINTR Indicates request interrupted by signal.
EPERM Indicates a permanent error occurred.
EBUSY Indicates device busy.
ENXIO Indicates unsupported device number.
ENODEV Indicates unsupported device or device type mismatch.

Files
/usr/include/sys/inputdd.h Contains declarations for ioctl commands and input ring report format.

Related Information
The GIO special file, kbd special file, lpfk special file, mouse special file, tablet special file.
The close subroutine, ioctl subroutine, open subroutine, read subroutine, write subroutine.
[Special Files Overview].

dump Special File

Purpose
Supports system dump.

Syntax
#include <sys/dump.h>

Description
The /dev/sysdump and /dev/sysdumpctl special files support system dumping. Minor device 0 of the sysdump driver provides the interfaces for the system dump routine to write data to the dump device. The sysdump driver also provides interfaces for querying or assigning the dump devices and initiating a dump.

Related Information
The dmp_ctl kernel service.
entn Special File

Purpose
Provides access to Ethernet high-performance LAN adapters by way of the Ethernet device handler.

Description
The /dev/entn character special file provides access to the Ethernet device handler for the purpose of providing access to an Ethernet LAN. The device handler supports up to four adapters, each of which may be running either or both of the standard Ethernet and IEEE 802.3 protocols.

Usage Considerations
When accessing the Ethernet device handler, the following should be taken into account:

Driver Initialization and Termination: The device handler can be loaded and unloaded. The handler supports the configuration calls to initialize and terminate itself.

Special File Support: Calls other than the open and close subroutines are discussed based on the mode in which the device handler is operating.

Subroutine Support
The Ethernet device handler supports the open and close, read, write and ioctl subroutines in the following manner:

open and close Subroutines: The device handler supports the /dev/entn special file as a character-multiplex special file. The special file must be opened for both reading and writing (O_RDWR). However, there are no particular considerations for closing the special file. The special file name used in an open call depends upon how the device is to be opened. Types of special file names are:

/dev/entn An open call to this device is used to start the device handler for the selected port, where the value of n is 0 <= n <= 7.
/dev/entn/D An open call to this device is used to start the device handler for the selected port in diagnostic mode, where the value of n is 0 <= n <= 7.

read Subroutine: Can take the form of a read, readx, readv, or readvx subroutine. For this call, the device handler copies the data into the buffer specified by the caller.

write Subroutine: Can take the form of a write, writex, writev, or writevx subroutine. For this call, the device handler copies the user data into a buffer and transmits the data on the LAN.

ioctl Subroutine: The Ethernet device handler supports the following ioctl operations:

<table>
<thead>
<tr>
<th>ioctl</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCC_GET_VPD</td>
<td>Returns adapter vital product data (VPD) if available and valid.</td>
</tr>
<tr>
<td>CIO_GET_FASTWRT</td>
<td>Returns the parameters required to issue an ent_fastwrt call.</td>
</tr>
<tr>
<td>CIO_GET_STAT</td>
<td>Returns current adapter and device handler status.</td>
</tr>
<tr>
<td>CIO_HALT</td>
<td>Halts a session and unregisters a network ID.</td>
</tr>
<tr>
<td>CIO_QUERY</td>
<td>Returns the current RAS counter values, as defined in the sys/comio.h and sys/entuser.h files.</td>
</tr>
<tr>
<td>CIO_START</td>
<td>Starts a session and registers a network ID.</td>
</tr>
<tr>
<td>ENT_SET_MULTI</td>
<td>Sets or clears a multicast address.</td>
</tr>
<tr>
<td>IOCINFO</td>
<td>Returns a device information structure to the user specified area. The devtype field value is DD_NET_DH and the devsubtype field is value DD_EN, as defined in the sys/devinfo.h file.</td>
</tr>
</tbody>
</table>
Error Codes
The following error codes may be returned when accessing the device handler through the `dev/entn` special file:

- **EACCES**
  Permission to access the port is denied for one of the following reasons:
  - The device has not been initialized.
  - The request to open the device in Diagnostic mode is denied.
  - The call is from a kernel mode process.

- **EAFNOSUPPORT**
  The address family is not supported by the protocol, or the multicast bit in the address is not set.

- **EAGAIN**
  The transmit queue is full.

- **EBUSY**
  The request is denied because the device is already opened in Diagnostic mode, or the maximum number of opens was reached.

- **EEXIST**
  The define device structure (DDS) already exists.

- **EFAULT**
  An address or parameter was specified that is not valid.

- **EINVAL**
  A range or operation code that is not valid was specified, or the device is not in Diagnostic mode.

- **EIO**
  An I/O error occurred.

- **ENOBDFS**
  No buffers are available.

- **ENOCONNET**
  A connection was not established.

- **ENODEV**
  The device does not exist.

- **ENOENT**
  There is no DDS to delete.

- **ENOMEM**
  The device does not have enough memory.

- **ENOMSG**
  No message of desired type was available.

- **ENOSPC**
  No space is left on the device (the multicast table is full).

- **ENOTREADY**
  The device is not ready, a CIO_START operation was not issued, or the operation was issued but did not complete.

- **ENXIO**
  The device does not exist, or the maximum number of adapters was exceeded.

- **EUNATCH**
  The protocol driver is not attached.

Related Information
The `close` subroutine, `open` subroutine, `read` or `readx` subroutine, `write` or `writex` subroutine, `ioctl` subroutine.

Error Logging Special Files

Purpose
Support error logging.

Description
The `error` and `errorctl` special files support the logging of error events. Minor device 0 (zero) of the `error` special file is the interface between processes that log error events and the `errdemon` (error daemon). Error records are written to the `error` special file by the `erlog` library routine and the `errsave` and `errlast` kernel services. The `error` special file timestamps each error record entry.

The error daemon opens `error` file for reading. Each read retrieves an entire error record. The format of error records is described in the `erec.h` header file.

Each time an error is logged, the error ID, the resource name, and the time stamp are recorded in nonvolatile random access memory (NVRAM). Therefore, in the event of a system crash, the last logged error is not lost. When the `error` file is restarted, the last error entry is retrieved from NVRAM.
The standard device driver interfaces (open, close, read, and write) are provided for the error file. The error file has no ioctl functions.

The ioctl function interface for the error special file is provided by the errorctl special file. This interface supports stopping the error logging system, synchronizing the error logging system, and querying the status of the error special file.

Related Information

- Special Files Overview in AIX 5L Version 5.3 Files Reference
- ioctl subroutine
- errclear command, errdead command, errdemon command, errinstall command, errlogger command, errmsg command, errpt command, errstop command, errupdate command.
- errlog subroutine.
- errsave and errlast kernel services.
- Error Logging Overview in AIX 5L Version 5.3 General Programming Concepts: Writing and Debugging Programs.

**fd Special File**

**Purpose**

Provides access to the diskette device driver.

**Description**

The fd special file provides block and character (raw) access to diskettes in the diskette drives. The special file name usually specifies both the drive number and the format of the diskette. The exceptions are /dev/fd0 and /dev/fd1, which specify diskette drives 0 and 1, respectively, without specifying their formats.

The generic special files /dev/fd0 and /dev/fd1 determine the diskette type automatically for both drive 0 and drive 1. First, the device-driver attempts to read the diskette using the characteristics of the default diskette for the drive type. If this fails, the device-driver changes its characteristics and attempts to read until it has read the diskette successfully or until it has tried all the possibilities supported for the drive type by the device driver.

An r prefix on a special file name means that the drive is accessed as a raw device rather than a block device. Performing raw I/O with a diskette requires that all data transfers be in multiples of the diskette sector length. Also, all lseek subroutine calls made to the raw diskette device driver must result in a file offset value that is a multiple of the sector size. For the diskette types supported, the sector length is always 512 bytes.

**Note:** The diskette device driver does not perform read verification of data that is written to a diskette.

**Types of Diskettes Supported**

The fd special file supports three diskette drives: the 1.2MB, 5.25-inch diskette drive, and the 1.44MB and 2.88MB, 3.5-inch diskette drives. All fd special file names (except the generic special files /dev/fd0, /dev/fd1, /dev/rfd0, and /dev/rfd1) contain suffixes that dictate how a diskette is to be treated. These special file names have a format of PrefixXY, where the Prefix, X, and Y variables have the following meanings:
Prefix Special file type. Possible values are \texttt{fd} and \texttt{rfd}, where the \texttt{r} indicates raw access to the special file.

\texttt{X} Drive number indicator. Possible values of 0 and 1 indicate drives 0 and 1, respectively.

\texttt{Y} Diskette format indicator. Possible values depend on the type of diskette being used. Either a single character or a decimal point followed by numeric characters is allowed. Possible values are:

- \texttt{h} Highest density supported by the drive type
- \texttt{l} Lowest density supported by the drive type
- \texttt{.9} 9 sectors per track (all three drive types)
- \texttt{.15} 15 sectors per track (1.2MB, 5.25-inch drive only)
- \texttt{.18} 18 sectors per track (both 3.5-inch drive types)
- \texttt{.36} 36 sectors per track (2.88MB, 3.5-inch drive only)

### 1.44MB, 3.5-inch Diskette Special Files:
Ten different special files are available for use with the 1.44MB, 3.5-inch diskette drive. The default diskette type assumed for this drive type is a double-sided, 80-cylinder, 18 sectors-per-track diskette.

An \texttt{h} or \texttt{.18} as the suffix of the special file name (for example, \texttt{/dev/rfd0h} or \texttt{/dev/fd0.18}) forces a diskette to be treated as a double-sided, 80-cylinder, 18 sectors-per-track diskette. An \texttt{l} or \texttt{.9} as the suffix of the special file name (for example, \texttt{/dev/fd1l} or \texttt{/dev/rfd0.9}) forces a diskette to be treated as a double-sided, 80-cylinder, 9 sectors-per-track diskette.

### 2.88MB, 3.5-inch Diskette Special Files:
Twelve different special files are available for use with the 2.88MB, 3.5-inch diskette drive. The default diskette type assumed for this drive type is a double-sided, 80-cylinder, 36 sectors-per-track diskette.

An \texttt{h} or \texttt{.36} as the suffix of the special file name (for example, \texttt{/dev/fd1h} or \texttt{/dev/fd0.36}) forces a diskette to be treated as a double-sided, 80-cylinder, 36 sectors-per-track diskette. An \texttt{l} or \texttt{.9} as the suffix of the special file name (for example, \texttt{/dev/rfd0l} or \texttt{/dev/fd1.9}) forces a diskette to be treated as a double-sided, 80-cylinder, 9 sectors-per-track diskette. A suffix of \texttt{.18} (for example, \texttt{/dev/fd1.18}) forces a diskette to be treated as a double-sided, 80-cylinder, 18 sectors-per-track diskette.

### 1.2MB, 5.25-inch Diskette Special Files:
Ten different special files are available for use with the 1.2MB, 5.25-inch diskette drive. The default diskette type assumed for this drive type is a double-sided, 80-cylinder, 15 sectors-per-track diskette.

An \texttt{h} or \texttt{.15} as the suffix of the special file name (for example, \texttt{/dev/rfd1h} or \texttt{/dev/fd0.15}) forces a diskette to be treated as a double-sided, 80-cylinder, 15 sectors-per-track diskette. An \texttt{l} or \texttt{.9} as the suffix of the special file name (for example, \texttt{/dev/fd0l} or \texttt{/dev/rfd1.9}) forces a diskette to be treated as a double-sided, 80-cylinder, 9 sectors-per-track diskette.

### Note:
Regardless of the diskette drive type, an \texttt{h} as the suffix of the special file name forces a diskette to be treated as the highest capacity diskette supported by the drive type. When an \texttt{l} is used as the suffix of the special file name, the diskette is treated as the lowest capacity diskette supported by the drive type.

### Usage Considerations
When using subroutines with the \texttt{fd} special file, consider the following items:

- **open and close subroutines**: Only one process at a time can issue an \texttt{open} subroutine to gain access to a particular drive. However, all child processes created by a parent process that successfully opens a diskette drive inherit the open diskette drive.

- **read and write subroutines**: No special considerations.
The possible `ioctl` operations and their descriptions are:

**IOCINFO**
Returns a `devinfo` structure (defined in the `/usr/include/sys/devinfo.h` file) that describes the device.

**FDIOCSINFO**
Sets the characteristics of the device driver diskette to the values passed in the `fdinfo` structure, as defined in the `/usr/include/sys/fd.h` file.

**FDIOCGINFO**
Gets the device-driver diskette characteristics and returns the values in the `fdinfo` structure, as defined in the `/usr/include/sys/fd.h` file.

**FDIOCFORMAT**
Formats a diskette track. The diskette is formatted using data passed in an array of bytes. The length of this array is four times the number of sectors per track on the diskette. The reason for this is that 4 bytes of data must be passed in for every sector on the track. The 4 bytes contain, in this order, the cylinder number, the side number (0 or 1), the sector number, and the number of bytes per sector. This pattern must be repeated for every sector on the track.

The diskette characteristics used during formatting are whatever values are in the device driver when it receives the format command. These characteristics need to be set to the desired values prior to issuing the `format` command. There are three ways to do this:

- Open the diskette driver using one of the format-specific special files. As a result, the diskette characteristics for the driver will be those of the diskette indicated by the special file.
- Open the diskette driver using one of the generic special files. In this case, the diskette characteristics will be the default characteristics for that driver.
- Set the characteristics explicitly using the `FDIOCSINFO` `ioctl` operation.

For formatting, the diskette driver should be opened only when the `O_NDELAY` flag is set. Otherwise, the driver will attempt to determine the type of diskette in the drive, causing the open to fail.

**Related Information**
The `close` subroutine, `ioctl` subroutine, `lseek` subroutine, `open` subroutine, `read` subroutine, `write` subroutine.

**fddin Special File**

**Purpose**
Provides access to the FDDI device driver by way of the FDDI device handler.
Description

The *fddin* special file provides access to the FDDI device handler that provides access to a FDDI local area network.

When accessing the FDDI device driver, the following information should be taken into account.

Driver Initialization and Termination

The device driver can be loaded and unloaded. The device driver supports the configuration calls to initialize and terminate itself.

Special File Support

Subroutine calls other than those made with the `open` and `close` subroutines are discussed based on the mode in which the device driver is operating.

Subroutine Support

The FDDI device driver provides specific support for the `open`, `close`, `read`, `write`, `ioctl`, `select`, and `poll` subroutines.

The device driver supports the `/dev/fddin` special file as a character-multiplex special file. The special file must be opened for both reading and writing. There are no particular considerations for closing the special file. The special file name used in an open call differs depending upon how the device is to be opened. Types of special file names are:

- `/dev/fddin` Starts the device driver for the selected port.
- `/dev/fddin/D` Starts the device driver for the selected port in Diagnostic mode.
- `/dev/fddin/C` Starts the device driver for the selected port in Diagnostic Configuration mode.

Error Codes

The following error conditions may be encountered when accessing the FDDI device driver through the `/dev/fddin` special file. The error codes can be found in the `/usr/include/sys/errno.h` file.

- **ENODEV** Indicates that an invalid minor number was specified.
- **EINVAL** Indicates that an invalid parameter was specified.
- **ENOMEM** Indicates that the device driver was unable to allocate the required memory.
- **EINTR** Indicates that a system call was interrupted.
- **EPERM** Indicates that the Diagnostic mode open request was denied because the device was already open.
- **EACCES** Indicates one of the following:
  - A non-privileged user tried to open the device in Diagnostic mode.
  - An illegal call from a kernel-mode user.
  - An illegal call from a user-mode user.
- **ENETDOWN** Indicates one of the following:
  - The network is down. The device is unable to process the requested operation.
  - An unrecoverable hardware error.
- **ENETUNREACH** Indicates that the device is in Network Recovery mode and is unable to process the requested operation.
- **ENOCONNECT** Indicates that the device has not been started.
- **EAGAIN** Indicates that the transmit queue is full.
- **EFAULT** Indicates that an invalid address was supplied.
- **EIO** Indicates an error. See the status field for detailed information.
- **EMSGSIZE** Indicates that the data was too large to fit into the receive buffer and that no `ext` parameter was supplied to provide an alternate means of reporting this error with a status of `CIO_BUF_OVFLW`.
Related Information
The close subroutine, ioctl subroutine, open subroutine, poll subroutine, read subroutine, select subroutine, and write subroutine.

Special Files Overview in AIX 5L Version 5.3 Files Reference.

GIO Special File

Purpose
Provides access to the graphics I/O (GIO) adapter.

Description
The GIO special file is the application interface to the GIO adapter. The GIO special file provides applications with the ability to determine what I/O devices are attached to the GIO adapter.

Configuration
Standard configuration methods are provided for the GIO special file. User configurable attributes for the GIO special file do not exist.

Usage Considerations
The open subroutine call specifying the GIO special file is processed normally except that the Oflag and Mode parameters are ignored. An open request is rejected if the special file is already opened or if a kernel extension attempts to open the GIO special file.

Calls to the read and write routines behave as if the call was made to the /dev/null file.

The GIO special file supports the following functions with ioctls:

- IOCINFO Returns the devinfo structure.
- GIOQUERYID Returns the identifier of device connected to the GIO adapter.

Error Codes
The following error codes can be found in the /usr/include/sys/errno.h file:

- EFAULT Indicates insufficient authority to access address or invalid address.
- EIO Indicates an I/O error.
- ENOMEM Indicates insufficient memory for required paging operation.
- ENOSPC Indicates insufficient file system or paging space.
- EINVAL Indicates that an invalid argument was specified.
- EINTR Indicates a request interrupted by signal.
- EPERM Indicates a permanent error occurred.
- EBUSY Indicates the device is busy.
- ENXIO Indicates an unsupported device number.
- ENODEV Indicates an unsupported device or device type mismatch occurred.

Files
/usr/include/sys/inputdd.h Contains the ioctl commands.
Related Information
The `close` subroutine, `ioctl` subroutine, `open` subroutine, `read` subroutine, `write` subroutine.

The `dials` special file, `lpfk` special file.

Special Files Overview.

**ide Special File**

**Purpose**
Provides access to the Integrated Device Electronics (IDE) adapter driver.

**Description**
The `ide` special file provides an interface to an attached IDE Bus. This special file should not be opened directly by application programs. The `/dev/ide0`, `/dev/ide1`, ... `/dev/ideN` files are the `ide` special files.

**Related Information**
Special Files Overview.


**kbd Special File**

**Purpose**
Provides access to the natively attached keyboard.

**Description**
The `kbd` special file is the interface to the native keyboard. It provides an interface through which applications can receive operator input from the keyboard, control the keyboard LED's, and changing various keyboard parameters. The special file also allows an application to send an audible signal to the operator via the speaker located within the keyboard.

**Configuration**
The sound volume, click volume, typematic rate and typematic delay are configurable by the application through the `ioctl` subroutine. These changes are not reflected in the ODM database. To change these attributes in the ODM database, use the `chhwkbd` command.

**Usage Considerations**

`open`
This subroutine call creates a channel between the application and the natively attached keyboard. Two channels are supported. The open subroutine call is processed normally except that the `MODE` and `Oflag` parameters are ignored. All keyboard input is flushed until an input ring is established. Only the input ring associated with the most recent open receives input reports.
close
When the kbd device has been opened twice, input is reported through the input ring registered previous to the first open, after the close subroutine call.

read and write
The keyboard device driver does not return nor accept data via read and write. These calls behave as if the call was made to /dev/null. Input data is received from the input drivers via the input ring.

ioctl
The keyboard device driver supports the following ioctl commands:

- IOCINFO: Return devinfo structure.
- KSAalarm: Sound alarm.
- KSCFGCLICK: Control keyboard click.
- KSdiagnostic: Enable/disable diagnostics mode (user mode only).
- KSLED: Set/reset keyboard LED's.
- KSKAP: Enable/disable keep alive poll (user mode only).
- KSKAPACK: Acknowledge keep alive poll (user mode only).
- KSQUERYID: Query keyboard device identifier.
- KQUERYSV: Query keyboard service vector (kernel mode only).
- KSREGRING: Register input ring.
- KSRFLUSH: Flush input ring.
- KSTDelay: Set typamatic delay.
- KSTRate: Set typamatic rate.
- KSVolume: Set alarm volume

Error Codes
The error codes can be found in the /usr/include/sys/errno.h file.

- EFAULT: indicates insufficient authority to access address or invalid address.
- EIO: indicates that an I/O error occurred.
- ENOMEM: indicates there was insufficient memory for required paging operation.
- ENOSPC: indicates there was insufficient file system or paging space.
- EINVAL: indicates that an invalid argument was specified.
- EINTR: indicates the request was interrupted by signal.
- EPERM: indicates that a permanent error occurred.
- EBUSY: indicates the device is busy.
- ENXIO: indicates unsupported device number was specified.
- ENODEV: indicates an unsupported device or device type mismatch.

Files
/usr/include/sys/inputdd.h Contains declarations for ioctl commands and input ring report format.

Related Information
The close subroutine, ioctl subroutine, open subroutine, read subroutine, write subroutine.

Special Files Overview.
Ift Special File

Purpose
Provides character-based terminal support for the local graphics display and keyboard.

Description
The Ift file is the application interface to the "Low Function Terminal (LFT) Subsystem". It provides support for a VT100-like terminal which is associated with the local graphics display and keyboard. It provides only character operations and is designed to be used during system installation, startup, shutdown, and stand-alone diagnostics.

The terminal supports a single logical screen size of 80 characters and 25 lines and a single color. Dynamic logical partitioning is not supported, configuration changes take effect at the next system startup. In the cases when multiple fonts may be used to achieve the 80x25 screen size, the user may set which font is used with the next system restart. See "LFT User Commands" for details of the available commands.

When multiple displays are available, the LFT Subsystem initially uses the default display. The user may change to another display and set the default display. See "LFT User Commands" for details of the available commands.

Usage Considerations
The LFT device driver supports the Ift special file. The device driver is a streams based driver. It handles only the system attached keyboard and graphics displays.

Sharing Displays with Graphic Subsystem
Certain LFT ioctl commands allow graphics subsystems to obtain exclusive use of the displays, a right initially held by the LFT. However, this is done by the Rendering Context Manager (RCM) on behalf of the graphics subsystem. See "Rendering Context Manager" for details of the procedure for becoming a graphics process.

Subroutine Support
The Ift special file supports the open, close, read, write, and ioctl subroutines.

ioctl system call: The functions performed by the ioctl commands fall into three categories:

- **Sharing devices** between the Ift and a graphic subsystem
- **Query information about configured displays and keyboard devices**
- **Compatibility with the common tty ioctl commands**

Sharing devices:

- IOCINFO The IOCINFO ioctl operation is defined for all device drivers that use the ioctl subroutine. The IOCINFO operation returns a devinfo structure, which is defined in the devinfo.h file.
- LFT_SET_DEFAULT_DISP Sets the default display.
- LFT_ACQ_DISP Acquire display for exclusive use.
- LFT_REL_DISP Release display.
- LFT_DIAG_OWNER Acquire display for diagnostics.

Query information about configured displays and keyboard devices:

- LFT_QUERY_LFT Query common LFT information.
- LFT_QUERY_DISP Query display information.
Compatibility with the common tty ioctl commands: TCSAK

TCGETA
TCSETAW
TCSETAF
TCSETA
TCSETAWS
TIOCGWINSZ
TIOCSWINSZ
TXTTYNAME
TSCBRK

Related Information


rcm and kbd Special Files.
Special Files Overview.

lp Special File

Purpose
Provides access to the line printer device driver.

Description
The lp driver provides an interface to the port used by a printer.

Printer Modes
The lp driver interprets carriage returns, backspaces, line feeds, tabs, and form feeds in accordance with the modes that are set in the driver (through the spip command or configuration). The number of lines per page, columns per line, and the indentation at the beginning of each line can also be selected. The default for these modes can be found using the lsattr command. The following modes can be set with the LPRMODS ioctl operation:

PLOT  Determines if the data stream is interpreted by the device driver when formatting the text. If the PLOT mode is off, the text is formatted using the current values set with the LPRSET ioctl operation.

If the PLOT mode is set, no interpretation of the data stream is performed and the bytes are sent to the printer without modification. Setting the PLOT mode causes other formatting modes, such as NOFF and NOFL, to be ignored. The default printer backend, piobe, sends all output in PLOT mode.

When in PLOT mode, the application must send a final form-feed character. If the last write operation was performed while not in PLOT mode, the final form-feed character will be sent by the device driver.
NOFF
If this mode is on, each form-feed character is replaced with a line-feed character, based on the current line value set with the `LPRSET` ioctl operation. This mode is ignored if the PLOT mode is active.

NONL
If this mode is on, each line-feed character is replaced with a carriage return. This mode is ignored if the PLOT mode is active.

NOCL
If this mode is off, a carriage return is inserted after each line-feed character. If the mode is on, no carriage return is inserted after the line-feed character. This mode is ignored if the PLOT mode is active.

NOTAB
If this mode off, 8 position tabs are simulated using spaces. If the NOTAB mode is on, the tab character is replaced with a space. This mode is ignored if the PLOT mode is active.

NOBS
If this mode off, backspaces are sent to the printers. If the NOBS mode is on, the backspace is simulated by sending a carriage return followed by spaces to the proper print position. This mode is ignored if the PLOT mode is active.

NOCR
If this mode on, each carriage return is replaced with a line-feed character. This mode is ignored if the PLOT mode is active.

CAPS
If this mode on, lowercase characters are converted to uppercase. This mode is ignored if the PLOT mode is active.

WRAP
If this mode off, the line is truncated at the right margin and any characters received past the right margin are discarded. If the WRAP mode is on, the characters received after the right margin are printed on the next line preceded by ... (ellipsis). This mode is ignored if the PLOT mode is active.

FONTINIT
The FONTINIT mode is initially off. It is turned on by an application when a printer font has been initialized. It can be turned off in the following two cases:

- An application needs fonts to be reinitialized.
- A fatal printer error occurs. In this case, the `lp` device driver turns the FONTINIT mode off.

RPTERR
If the RPTERR mode is off and an error occurs, the device driver does not return until the error has been cleared or a cancel signal is received. If the RPTERR mode is on, the device driver waits the amount of time specified in the `v_timeout` configuration parameter and then returns with an error.

IGNOREPE
If IGNOREPE mode is on, the device driver allows writes to the device regardless of the state of the PE (paper-end) line on the parallel interface. An application can make use of this mode, for example, to change the paper tray of a printer under software control when detecting that the printer is out of paper.

Error Handling When the RPTERR Mode Is Off
If the RPTERR mode is off, no error reporting is performed. The device driver waits for the error to be cleared or a cancel signal to be received before returning to the application. RPTERR is the default mode and is intended for existing applications that do not perform error recovery.

If a signal is received by the device driver, the current operation is returned incomplete with an `EINTR` error code.

If printing is canceled and the printer is in PLOT mode, it is the application must send the final form-feed character to eject the partial page. If the printer is not in PLOT mode, the final form-feed character after cancelation will be sent by the device driver.

Error Handling When the RPTERR Mode Is On
If the RPTERR mode is on, the device driver will wait for the time specified in the `v_timeout` configuration parameter and then return the uncompleted operation with an error code. This return allows the application to get the printer status and possibly display an error message.

Note: When a device driver returns an incomplete operation with an error code (as previously described), the application must resend any data not printed.
Usage Considerations

Device-Dependent Subroutines: Most printer operations are implemented using the open, read, write, and close subroutines. However, these subroutines provide little or no information to the calling program about the configuration and state of the printer. The ioctl subroutine provides a more device-specific interface to the printer device driver.

Most of these subroutines pass data contained in structures. In all cases, a structure of the type indicated should be allocated in the calling routine. A pointer to this structure should then be passed to the device driver.

open and close Subroutines: If an adapter for a printer is not installed, an attempt to open fails. If the printer adapter is busy, the open subroutine returns an error. However, all child processes created by a parent process that successfully opens the lp special file inherit the open printer.

The driver allows multiple open subroutines to occur if they all have a mode parameter value of read-only. Thus, the splp command can perform inquiries when the printer adapter is currently in use. The lp driver allows only one process to write to a printer adapter at a time.

The close subroutine waits until all output completes before returning to the user.

read and write Subroutines: The read subroutine is not implemented for the native I/O parallel port.

When printing to a parallel printer that is offline, the write subroutine may return one fewer than the actual number of bytes that are buffered and ready to be written when the printer is put back online. This is used as a mechanism to indicate to the calling application that there is a problem with the printer requiring user intervention, possibly allowing the user to put the printer online and continue with printing. In this situation, no error is returned by the write subroutine.

ioctl Subroutine: The possible ioctl operations and their descriptions are:

- **IOCINFO**: Returns a structure defined in the /usr/include/sys/devinfo.h file, which describes the device.
- **LPQUERY**: Provides access to the printer status. Refer to the /usr/include/sys/lpio.h file for value definitions. The types of errors are the following:
  - The printer is out of paper.
  - No select bit: the printer may be turned off or not installed.
  - The printer is busy.
  - The printer is unknown.
- **LPRGET**: Returns the page length, width and indentation values. These values are used by the device driver when PLOT mode is not set. The default printer backend, piobe, sends all print jobs with PLOT mode set. The LPRGET operation uses the lprio structure, as defined in the /usr/include/sys/lpio.h file.
- **LPRGETA**: Gets the RS232 parameters. These are the values for baud rate, character rate, character size, stop bits and parity. Refer to the LPR232 structure and to the termio structure, as defined in the termios.h file.
  - **Note**: This operation is supported for compatibility reasons. The use of the tcgetatt attr subroutine is recommended.
- **LPRGTOV**: Gets the current time-out value and stores it in the lptimer structure defined in the /usr/include/sys/lpio.h file. The time-out value is measured in seconds.
- **LPRMODO**: Gets the printer modes. These printer modes support the various formatting options and error reporting. This ioctl operation uses the LPRMOD structure, as defined in the /usr/include/sys/lpio.h file.
- **LPRMODS**: Sets the printer modes. These printer modes support the various formatting options and error reporting. This ioctl operation uses the LPRMOD structure, as defined in the /usr/include/sys/lpio.h file.
- **LPRSET**: Sets the page length, width and indent values. These values are used by the device driver when PLOT mode is not set. The default printer backend, piobe, sends all print jobs with PLOT mode set. The LPRSET operation uses the lprio structure, as defined in the /usr/include/sys/lpio.h file.
LPRSETA  Sets the RS232 parameters. These are the values for baud rate, character rate, character size, stop
bits and parity. Refer to the LPR232 structure and to the termio structure, as defined in the termios.h
header file.

Note: This operation is supported for compatibility reasons. The use of the tcsetattr subroutine is
recommended.

LPRSTOV  Sets the time-out value. The arg parameter to this ioctl operation points to a lptimer structure defined
in the /usr/include/sys/lpio.h file. The time-out value must be given in seconds.

Related Information

Printer Addition Management Subsystem: Programming Overview in AIX 5L Version 5.3 Kernel Extensions
and Device Support Programming Concepts

The lsattr command, piobe command, splp command.

The close subroutine, ioctl subroutine, open subroutine, read subroutine, write subroutine.

lpfk Special File

Purpose
Provides access to the lighted program function key (LPFK) array.

Description
The lpfk special file is the application interface to the lighted program function keys. It allows the
application to receive operator input from the LPFKs and to illuminate and darken each key in the array.

Configuration
Standard configuration methods are provided for the lpfk special file. The user cannot enter configurable
attributes by way of the command line.

Usage Considerations

open: An open subroutine call specifying the lpfk special file is processed normally except that the Oflag
and Mode parameters are ignored. An open request is rejected if the special file is already opened or if a
kernel extension attempts to open the lpfk special file. All LPFK inputs are flushed following an open call
until an input ring is established.

read and write: The lpfk special file does not support read or write subroutine calls. Instead, input data
is obtained from the LPFKs through the input ring. The read and write subroutine calls behave the same
as read and write functions of the /dev/null file. See “LFT Input Ring” in AIX 5L Version 5.3 Kernel
Extensions and Device Support Programming Concepts for how to use the input ring.

ioctl: The lpfk special file supports the following ioctl operations:

IOCINFO          Returns the devinfo structure.
LPFKREGRING      Registers input ring.
LPFKRFLUSH       Flushes input ring.
LPFKLIGHT        Sets key lights.
Error Codes
The error codes can be found in the /usr/include/sys/errno.h file.

**EFAULT** Indicates insufficient authority to access address, or invalid address.
**EIO** Indicates I/O error.
**ENOMEM** Indicates insufficient memory for required paging operation.
**ENOSPC** Indicates insufficient file system or paging space.
**EINVAL** Indicates invalid argument specified.
**EINTR** Indicates request interrupted by signal.
**EPERM** Indicates a permanent error occurred.
**EBUSY** Indicates device busy.
**ENXIO** Indicates unsupported device number.
**ENODEV** Indicates unsupported device, or device type mismatch.

Files
/usr/include/sys/inputdd.h Contains declarations for ioctl commands and input ring report format

Related Information
The dials special file, GIO special file, kbd special file, mouse special file, and tablet special file.
The close subroutine, ioctl subroutine, open subroutine, read subroutine, write subroutine.

Special Files Overview.

lvdd Special File
Purpose
Provides access to the logical volume device driver.

Description
The logical volume device driver provides character (raw) access to logical volumes. The Logical Volume Manager associates a major number with each volume group and a minor number with each logical volume in a volume group.

Logical volume special file names can be assigned by the administrator of the system. However, /dev/lv1, /dev/lv2 and /dev/rlv1, /dev/rlv2 are the names conventionally chosen.

When performing character I/O, each request must start on a logical block boundary of the logical volume. The logical block size is 512 bytes. This means that for character I/O to a logical volume device, the offset supplied to the seek subroutine must specify a multiple of 512 bytes. In addition, the number of bytes to be read or written, supplied to the read or write subroutine, must be a multiple of 512 bytes.

Note: I/O requests should not be sent to the block special file interface when the logical volume is mounted. When a logical volume is mounted (that is, the block special file is opened by the file system), any I/O requests from the user made to that logical volume should be made only through the character special file.
Usage Considerations

**Note:** Data corruption, loss of data, or loss of system integrity (system crashes) will occur if devices supporting paging, logical volumes, or mounted file systems are accessed using block special files. Block special files are provided for logical volumes and disk devices on the operating system and are solely for system use in managing file systems, paging devices and logical volumes. They should not be used for other purposes. Additional information concerning the use of special files may be obtained in "Understanding I/O Access through Special Files" in *AIX 5L Version 5.3 Kernel Extensions and Device Support Programming Concepts*.

open and close Subroutines: No special considerations.

**Extension Word Specification for the readx and writex Subroutines:** The ext parameter for the readx and writex extended I/O subroutines indicates specific physical or logical operations, or both. The upper 4 bits of the ext parameter are reserved for internal LVDD use. The value of the ext parameter is defined by logically ORing values from the following list, as defined in the `/usr/include/sys/lvdd.h` file:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WRITEV</td>
<td>Perform physical write verification on this request. This operation can be used only with the writex subroutine.</td>
</tr>
<tr>
<td>RORELOC</td>
<td>For this request, perform relocation on existing relocated defects only. Newly detected defects should not be relocated.</td>
</tr>
<tr>
<td>MWC_RCV_OP</td>
<td>Mirror-write-consistency recovery operation. This option is used by the recovery software to make consistent all mirrors with writes outstanding at the time of the crash.</td>
</tr>
<tr>
<td>NOMWC</td>
<td>Inhibit mirror-write-consistency recovery for this request only. This operation can only be used with the writex subroutine.</td>
</tr>
<tr>
<td>AVOID_C1, AVOID_C2, AVOID_C3</td>
<td>For this request, avoid the specified mirror. This operation can only be used with the readx subroutine.</td>
</tr>
<tr>
<td>RESYNC_OP</td>
<td>For this request, synchronize the specified logical track group (LTG). This operation can only be used with the readx subroutine and must be the only operation. When synchronizing a striped logical volume, the data returned is not usable by the application because the logical track group is not read on a striped basis.</td>
</tr>
<tr>
<td>LV_READ_BACKUP</td>
<td>Read only the mirror copy that is designated as the backup mirror copy.</td>
</tr>
<tr>
<td>LV_WRITE_BACKUP</td>
<td>Write only the mirror copy that is designated as the backup mirror copy.</td>
</tr>
<tr>
<td>LV_READ_ONLY_C1</td>
<td>Read only copy one of the data.</td>
</tr>
<tr>
<td>LV_READ_ONLY_C2</td>
<td>Read only copy two of the data.</td>
</tr>
<tr>
<td>LV_READ_ONLY_C3</td>
<td>Read only copy three of the data.</td>
</tr>
<tr>
<td>LV_READ_STALE_C1</td>
<td>Read only copy one of the data even if it is stale.</td>
</tr>
<tr>
<td>LV_READ_STALE_C2</td>
<td>Read only copy two of the data even if it is stale.</td>
</tr>
<tr>
<td>LV_READ_STALE_C3</td>
<td>Read only copy three of the data even if it is stale.</td>
</tr>
</tbody>
</table>

There are some restrictions when using this operation. To synchronize a whole logical partition (LP), a series of readx subroutines using the RESYNC_OP operation must be done. The series must start with the first logical track group (LTG) in the partition and proceed sequentially to the last LTG. Any deviation from this will result in an error. The length provided to each readx operation must be exactly 128KB (the LTG size).

Normal I/O can be done concurrently anywhere in the logical partition while the RESYNC_OP is in progress. If an error is returned, the series must be restarted from the first LTG. An error is returned only if resynchronization fails for every stale physical partition copy of any logical partition. Therefore, stale physical partitions are still possible at the end of synchronizing an LP.
Normal I/O operations do not need to supply the ext parameter and can use the read and write subroutines.

**IOCINFO ioctl Operation:** The IOCINFO ioctl operation returns the devinfo structure, as defined in the /usr/include/sys/devinfo.h file. The values returned in this structure are defined as follows for requests to the logical volume device driver:

- **devtype**
  - Equal to DD_DISK (as defined in the devinfo.h file)
- **flags**
  - Equal to DF_RAND
- **devsubtype**
  - Equal to DS_LV or DS_LVZ. The DS_LVZ devsubtype indicates that the logical volume control block will not occupy the first block of the logical volume, therefore, the space is available for application data. For oldvg format volume groups, the devsubtype of a logical volume is always DS_LV. For bigvg format volume groups, the devsubtype of a logical volume will be DS_LVZ if mklv -T 0 was used to create the logical volume. For scalable format volume groups, the devsubtype of a logical volume is always DS_LVZ (regardless of whether or not the mklv -T 0 flag was used to create the logical volume).
- **bytpsec**
  - Bytes per block for the logical volume
- **secptrk**
  - Number of blocks per logical track group
- **trkpcyl**
  - Number of logical track groups per partition
- **numblks**
  - Number of logical blocks in the logical volume

**XLATE ioctl Operation:** The XLATE ioctl operation translates a logical address (logical block number and mirror number) to a physical address (physical device and physical block number on that device). The caller supplies the logical block number and mirror number in the xlate_arg structure, as defined in the /usr/include/sys/lvdd.h file. This structure contains the following fields:

- **lbn**
  - Logical block number to translate
- **mirror**
  - The number of the copy for which to return a pbn (physical block number on disk). Possible values are:
  - 1: Copy 1 (primary)
  - 2: Copy 2 (secondary)
  - 3: Copy 3 (tertiary)
- **p_devte**
  - Physical dev_t (major/minor number of the disk)
- **pbn**
  - Physical block number on disk

**XLATE64 ioctl Operation:** The XLATE64 ioctl operation functions the same as the XLATE operation except that it uses the xlate_arg64 structure, in which the logical and physical block numbers and the device (major/minor) number fields are 64-bit wide.

**PBUFCNT ioctl Operation:** The PBUFCNT ioctl operation increases the size of the physical buffer header, pbuf, pool that is used by LVM for logical-to-physical request translation. The size of this pool is determined by the number of active disks in the system, although the pool is shared for request to all disks.

The PBUFCNT ioctl operation can be issued to any active volume group special file, for example /dev/VolGrpName. The parameter passed to this ioctl is a pointer to an unsigned integer that contains the pbufs-per-disk value. The valid range is 16 - 128. The default value is 16. This value can only be increased and is reset to the default at IPL. The size of the pbuf pool is not reduced when the number of active disks in the system is decreased.

The PBUFCNT ioctl operation returns the following:

- **EINVAL**
  - Indicates an invalid parameter value. The value is larger than the maximum allowed, or smaller than or equal to the current value.
- **EFAULT**
  - Indicates that the copy in of the parameter failed.
- **LVDD_ERROR**
  - An error occurred in allocating space for additional buffer headers.
- **LVDD_SUCCESS**
  - Indicates a successful ioctl operation.
**LV_INFO ioctl Operation:** The LV_INFO ioctl operation returns information about the logical volume in question. This ioctl operation only applies to AIX 4.2.1 and later.

The caller supplies the logical volume special file in the system open call and the information is returned via the `lv_info` structure, as defined in the `/usr/include/sys/lvdd.h` file. This structure contains the following fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vg_id</td>
<td>Volume group ID of which the logical volume is a member</td>
</tr>
<tr>
<td>major_num</td>
<td>Major number of logical volume</td>
</tr>
<tr>
<td>minor_num</td>
<td>Minor number of the logical volume</td>
</tr>
<tr>
<td>max_lps</td>
<td>Maximum number of logical partitions allowed for this logical volume</td>
</tr>
<tr>
<td>current_lps</td>
<td>Current size of the logical volume in terms of logical partitions</td>
</tr>
<tr>
<td>mirror_policy</td>
<td>Specifies the type of mirroring, if the logical volume is mirrored. Valid values are parallel, sequential, striped, and striped_parallel.</td>
</tr>
<tr>
<td>permissions</td>
<td>Specifies whether the logical volume is read only or read-write</td>
</tr>
<tr>
<td>bb_relocation</td>
<td>Specifies whether bad block relocation is activated for the logical volume</td>
</tr>
<tr>
<td>write_verify</td>
<td>Specifies whether the write verify command for writes to the logical volume is enforced</td>
</tr>
<tr>
<td>num_blocks</td>
<td>Number of 512 byte blocks that make up the logical volume. This value does not include mirrored logical volumes</td>
</tr>
<tr>
<td>mwcc</td>
<td>Specifies which mirrored write consistency check algorithm is set, if it is active.</td>
</tr>
<tr>
<td></td>
<td><strong>MWCC_NON_ACTIVE</strong></td>
</tr>
<tr>
<td></td>
<td>mwcc disabled for this logical volume</td>
</tr>
<tr>
<td></td>
<td><strong>MWCC_ACTIVE_MODE</strong></td>
</tr>
<tr>
<td></td>
<td>ACTIVE mwcc algorithm set for this logical volume</td>
</tr>
<tr>
<td></td>
<td><strong>MWCC_PASSIVE_MODE</strong></td>
</tr>
<tr>
<td></td>
<td>PASSIVE mwcc algorithm set for this logical volume</td>
</tr>
<tr>
<td></td>
<td><strong>MWCC_PASSIVE_RECOVERY</strong></td>
</tr>
<tr>
<td></td>
<td>logical mirrors undergoing PASSIVE mwcc recovery after system interruption</td>
</tr>
<tr>
<td>mirr_able</td>
<td>Specifies whether the logical volume is capable of being mirrored</td>
</tr>
<tr>
<td>num_mirrors</td>
<td>Number of mirror copies for this logical volume</td>
</tr>
<tr>
<td>striping_width</td>
<td>Number of drives across which this logical volume is striped</td>
</tr>
<tr>
<td>stripe_exp</td>
<td>Stripe block exponent value</td>
</tr>
<tr>
<td>backup_mirror</td>
<td>Backup mirror mask will be zero indicating there is not a backup copy active.</td>
</tr>
<tr>
<td></td>
<td><strong>AVOID_C1</strong></td>
</tr>
<tr>
<td></td>
<td>For the first copy</td>
</tr>
<tr>
<td></td>
<td><strong>AVOID_C2</strong></td>
</tr>
<tr>
<td></td>
<td>For the second copy</td>
</tr>
<tr>
<td></td>
<td><strong>AVOID_C3</strong></td>
</tr>
<tr>
<td></td>
<td>For the third copy</td>
</tr>
</tbody>
</table>

The LV_INFO ioctl operation returns the following:

**EFAULT** Indicates that the copy of the parameter failed.

**EFAULT** Indicates that the copy of the parameter failed.

LVM ioctl Operations Used to Modify Single Logical Volumes

**LV_QRYBKPCOPY** Query for designated backup mirror copy.

**LV_SETBKPCOPY** Designate backup mirror copy.

**LV_FSETBKPCOPY** Force new designation for backup mirror copy. Used when there are stale partitions on either the active mirror or backup mirror.
SET_SYNC_ON_RD: Causes the logical volume to go into MWCC_PASSIVE_RECOVERY mode. All reads from one mirror copy will cause non-read mirror copies to undergo a sync write.

CLR_SYNC_ON_RD: Clears the MWCC_PASSIVE_RECOVERY mode of the logical volume, if it exists. This clear should not be exercised if mirror consistency is not guaranteed.

LV_INFO64 ioctl Operation: The LV_INFO64 ioctl operation functions the same as the LV_INFO operation except that it uses the lv_info64 structure, in which the major_num and minor_num fields are 32-bit wide each and the num_blocks field is 64-bit wide.

FORCEOFF_VG ioctl Operation: You can force a volume group offline by the FORCEOFF_VG ioctl operation. You can issue this operation to any active volume group special file, for example the /dev/VolGrpName file. The parameter passed to this ioctl is a pointer to an integer that contains the value FORCE_VG_OFF as defined in the /usr/include/sys/lvdd.h file.

When you force the volume group offline, subsequent logical volume opens, and I/O requests and changes for volume group configuration fail. You must vary off the volume group and vary it on again to clear the forced offline condition.

If this operation is completed, a value of 0 is returned. Otherwise, a value of -1 is returned and the errno global variable is set to one of the following values:

-efault Indicates that the copy-in of the parameter fails.
-INVAL Indicates the parameter value is not valid.

Error Codes
In addition to the possible general errors returned by the ioctl subroutine, the following errors can also be returned from specific ioctl operation types.

ENXIO The logical volume does not exist. (This error type is relevant to the IOCINFO, Xlate ioctl, and Xlate64 operations.)
ENXIO The logical block number is larger than the logical volume size. (This error type is relevant only to the Xlate ioctl and Xlate64 ioctl operations.)
ENXIO The copy (mirror) number is less than 1 or greater than the number of actual copies. (This error type is relevant only to the Xlate ioctl and Xlate64 ioctl operations.)
ENXIO No physical partition has been allocated to this copy (mirror). (This error type is relevant only to the Xlate ioctl and Xlate64 ioctl operations.)

Related Information
The close subroutine, ioctl subroutine, seek subroutine, open subroutine, read subroutine, write subroutine.

Logical volume storage in Operating system and device management.

mem or kmem Special File

Purpose
Provides privileged virtual memory read and write access.

Description
Note: When incorrect access to virtual memory is made through these files, process termination, a system crash, or loss of system data integrity can result.
The /dev/mem and /dev/kmem character special files provide access to a pseudo device driver that allows read and write access to system memory or I/O address space. Typically, these special files are used by operating system utilities and commands (such as `sar`, `iostat`, and `vmstat`) to obtain status and statistical information about the system.

**Note:** Programs accessing these special files must have appropriate privilege. Commercial application programs should avoid using the /dev/mem and /dev/kmem files, since the virtual memory image is quite specific to the operating system level and machine platform. Use of these special files thus seriously affects the portability of the application program to other systems.

**Usage Considerations**

### kmem Special File Access:

The kmem special file provides access to the virtual memory address space for the current process, as it is seen by the kernel. The seek offset, set by the `lseek` subroutine, is used to specify the virtual address targeted for the read or write. The kmem pseudo-device driver only supports the `open`, `close`, `read`, `readx`, `writex`, and `write` subroutines.

The `knlist` system subroutine is typically used to obtain the addresses of kernel symbols to read or write through access provided by the kmem special file.

Before issuing a read or write operation, the `lseek` subroutine must be used to designate the relevant starting address in virtual memory. If this address is within the first two gigabytes of address space, then the `read` or `write` subroutine calls can be used. However, if the upper two gigabytes of address space are to be accessed, the `readx` and `writex` form of the subroutine calls must be used. In this case, the `ext` (extension) parameter must be set to a value of True. This causes the `lseek` offset to be interpreted relative to the upper 2 gigabytes of address space.

**Note:** The process address space is defined as shown in the Implementation of mem Special File Access section. This address space layout can vary on other machine platforms and versions of the operating system.

### mem Special File Access:

**Note:** Use of this special file by application programs should be strictly avoided, as it is provided for diagnostic and problem determination procedures only.

The mem special file access is specific to the system on which the operating system is running.

Please refer to the Implementation of mem Special File Access section for details on the function provided by this special file.

### Process Address Space Regions for the /dev/kmem Special File

The "Process Address Space Map" illustrates the layout of process address space regions as accessed through the /dev/kmem special file on this system.

<table>
<thead>
<tr>
<th>Lower 2 gigabytes of address space: Use read or write subroutines</th>
<th>Upper 4 bits of lseek offset</th>
<th>Process Address Space Regions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>Primary Kernel Region</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>User Text Region</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Process Private Region</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>Attached Data Mapped Files Region</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>Attached Data Mapped Files Region</td>
</tr>
</tbody>
</table>
Implementation of mem Special File Access

The **mem** special file has traditionally provided direct access to physical memory. This capability and its interface requirements are machine-specific. However, for this operating system this function is indirectly provided by using the *ext* (extension) parameter on the **readx** and **writex** subroutine calls. When a **readx** or **writex** subroutine call associated with the `/dev/mem` special file is issued, the *ext* parameter must contain a valid segment register value as defined in the *POWERstation and POWERserver Hardware Technical Reference - General Information* documentation for the platform type(s) on which the program will be run. This allows the program to access all physical memory mapped by the page table as well as the platform-specific I/O (T=1) segments.

The seek offset set by the **lseek** subroutine call is used to specify the address offset within the segment described by the *ext* parameter. The upper four bits of the offset are not used. The pseudo-device driver only supports the **open**, **close**, **read**, **readx**, **write**, and **writex** subroutine calls. The **lseek** subroutine call must also be used before the **readx** or **writex** subroutine calls are issued, in order to specify the address offset.

If a **read** or **write** subroutine call is used with this special file, the access to memory is identical to that provided by the `/dev/kmem` special file.

The **mem** special file is part of Base Operating System (BOS) Runtime.

**Files**

* /dev/mem Provides privileged virtual memory read and write access.
* /dev/kmem Provides privileged virtual memory read and write access.

**Related Information**

The **iostat** command, **sar** command, **vmstat** command.

The **close** subroutine, **ioctl** subroutine, **knlist** subroutine, **lseek** subroutine, **open** subroutine, **poll** subroutine, **read** subroutine, **select** subroutine, **write** subroutine.

Special Files Overview.
**mouse Special File**

**Purpose**
Provides access to the natively attached mouse.

**Description**
The `mouse` special file serves as an interface between the application interface and the system mouse. This special file provides the application with the ability to receive input from the mouse and allows the application to change mouse configuration parameters, such as mouse sampling rates and resolution.

**Configuration**
Standard configuration methods work with the mouse special file. No user configurable attribute commands exist for this special file. Applications that open the special file can modify device attribute dynamically using the appropriate ioctl interface; however, modifications are not saved in the configuration database.

**Usage Considerations**
The `open` subroutine call specifying the `mouse` special file is processed normally except that the *Oflag* and *Mode* parameters are ignored. The `open` request is rejected when the special file is already opened or when a kernel extension attempts to open the special file. All mouse inputs are flushed following an `open` subroutine call until an input ring is established. The mouse device is reset to the default configuration when an open request is made.

The `mouse` special file does not support the `read` or `write` subroutine calls. Instead, input data is obtained from the mouse via the input ring. The `read` and `write` subroutine calls behave the same as `read` or `write` to the `/dev/null` file.

The `mouse` special file supports the following functions with ioctls:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOCINFO</td>
<td>Returns a devinfo structure.</td>
</tr>
<tr>
<td>MQUERYID</td>
<td>Returns the query mouse device identifier.</td>
</tr>
<tr>
<td>MREGRING</td>
<td>Specifies the address of the input ring and the value to be used as the source identifier when enqueuing reports on the ring.</td>
</tr>
<tr>
<td>MRFLUSH</td>
<td>Flushes the input ring.</td>
</tr>
<tr>
<td>MTHRESHOLD</td>
<td>Sets the mouse reporting threshold.</td>
</tr>
<tr>
<td>MRESOLUTION</td>
<td>Sets the mouse resolution.</td>
</tr>
<tr>
<td>MSCALE</td>
<td>Sets the mouse scale factor.</td>
</tr>
<tr>
<td>MSAMPLERATE</td>
<td>Sets the mouse sample rate.</td>
</tr>
</tbody>
</table>

**Error Codes**
The following error codes can be found in the `/usr/include/sys/errno.h` file:

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EFAULT</td>
<td>Indicates insufficient authority to access an address or invalid address.</td>
</tr>
<tr>
<td>EIO</td>
<td>Indicates I/O error.</td>
</tr>
<tr>
<td>ENOMEM</td>
<td>Indicates insufficient memory for required paging operation.</td>
</tr>
<tr>
<td>ENOSPC</td>
<td>Indicates insufficient file system or paging space.</td>
</tr>
<tr>
<td>EINVAL</td>
<td>Indicates invalid argument specified.</td>
</tr>
<tr>
<td>EINVAL</td>
<td>Indicates that the request has been interrupted by a signal.</td>
</tr>
<tr>
<td>EPERM</td>
<td>Specifies a permanent error occurred.</td>
</tr>
<tr>
<td>EBUSY</td>
<td>Indicates a device is busy.</td>
</tr>
<tr>
<td>ENXIO</td>
<td>Indicates an unsupported device number.</td>
</tr>
<tr>
<td>ENODEV</td>
<td>Indicates an unsupported device or device type mismatch.</td>
</tr>
</tbody>
</table>
EACCES  Indicates that an open is not allowed.

Files
/usr/include/sys/inputdd.h Contains the ioctl commands.

Related Information
The close subroutine, ioctl subroutine, open subroutine, read subroutine, write subroutine.

mpcn Special File

Purpose
Provides access to the HDLC network device driver by way of the SDLC COMIO device driver emulator. This special file only applies to AIX 4.2.1 and later.

Description
The /dev/mpcn character special file provides access to the HDLC network device driver via the SDLC COMIO device driver emulator in order to provide access to a synchronous network. The SDLC COMIO emulator device handler supports multiple HDLC network devices.

Usage Considerations
When accessing the SDLC COMIO emulator device handler, consider the following information.

Driver Initialization and Termination
The device handler can be loaded and unloaded. The handler supports the configuration calls to initialize and terminate itself.

Special File Support
The SDLC COMIO emulator device handler uses the t_start_dev and t_chg_parms structures defined in the /usr/include/sys/mpqp.h file to preserve compatibility with the existing GDLC, MPQP API, and SNA Services interface. However, only a subset of the #define values are supported for the following t_start_dev structure fields:

phys_link  Indicates the physical link protocol. Only one type of physical link is valid at a time. The SDLC COMIO emulator device handler supports PL_232D (EIA-232D), PL_422A (EIA-422A/v.36), PL_V35 (V.35), PL_X21 (X.21 leased only), and PL_V25 (V.25bis EIA-422A autodial).

data_proto  Identifies the data protocol. The SDLC COMIO emulator device handler supports only the SDLC DATA_PRO_SDLC_HDX (half duplex) and the DATA_PRO_SDLC_FDX (full duplex) values.

baud_rate  Specifies the baud rate for transmit and receive clocks. The SDLC COMIO emulator device handler supports only external clocking where the DCE supplies the clock, and this field should be set to zero.

Subroutine Support
The SDLC COMIO emulator device handler supports the open, close, read, write, and ioctl subroutines in the following manner:

open and close Subroutines:  The device handler supports the /dev/mpcn special file as a character-multiplex special file. The special file must be opened for both reading and writing (O_RDWR). No special considerations exist for closing the special file.
**read Subroutine:** Can take the form of a `read`, `readx`, `readv`, or `readvx` subroutine call. For this call, the device handler copies the user data in to the buffer specified by the caller.

**write Subroutine:** Can take the form of a `write`, `writex`, `writev`, or `writevx` subroutine call. For this call, the device handler copies the user data into a buffer and transmits the data on the wide area network using the HDLC network device driver.

**ioctl Subroutine:** The `ioctl` subroutine supports the following flags:

- **CIO_START** Starts a session and registers a network ID.
- **CIO_HALT** Halts a session and removes a network ID.
- **CIO_QUERY** Returns the current reliability, availability, and serviceability (RAS) counter values. These values are defined in the `/usr/include/sys/comio.h` file.
- **MP_CHG_PARMS** Permits the data link control (DLC) to change certain profile parameters after the SDLC COMIO device driver emulator is started.

**Error Codes**

The following error codes can be returned when gaining access to the device handler through the `/dev/mpcn` special file:

- **ECHRNG** Indicates that the channel number is out of range.
- **EAGAIN** Indicates that the device handler cannot transmit data because of a lack of system resources, or, because an error returned from the HDLC network device driver's transmit routine.
- **EBUSY** Indicates that the device handler is already in use (opened/started) by another user.
- **EIO** Indicates that the handler cannot queue the request to the adapter.
- **EFAULT** Indicates that the cross-memory copy service failed.
- **EINVAL** Indicates one of the following:
  - The port is not set up properly.
  - The handler cannot set up structures for `write`.
  - The port is not valid.
  - A kernel process called a `select` operation.
  - The specified physical-link parameter is not valid for that port.
  - A kernel process called a `read` operation.
- **ENOMEM** Indicates one of the following:
  - No mbuf or mbuf clusters are available.
  - The total data length is more than one page.
  - There is no memory for internal structures.
- **ENOMSG** Indicates that the status-queue pointer is null, and there are no entries.
- **ENOTREADY** Indicates that the port state in the define device structure (DDS) is not in Data Transfer mode or that the implicit halt of port failed.
- **ENXIO** Indicates one of the following:
  - The port was not started successfully.
  - The channel number is illegal.
  - The driver control block pointer is null or does not exist.

This file functions with the SDLC COMIO emulator device handler over the HDLC network device driver. It emulates the SDLC API (full and half duplex) of the Multiprotocol Quad Port (MPQP) device handler.
mpqi Special File

Purpose
Provides access to the Multiport Model 2 Adapter (MM2) device driver via SNA Services, GDLC, or user-written applications compatible with current MPQP Applications Programming Interface (API).

Description
The Multiport Model 2 device driver provides access to the mpqi special file through SNA Services, Generic Data Link Control, or through user-written applications.

Usage Considerations
When accessing the Multiport Model 2 device driver via these methods, consider the following information:

Driver Initialization and Termination: The device driver can be loaded and unloaded in the kernel in the same way as other communications device drivers. The device driver supports the configuration calls to initialize and terminate itself. Therefore, you must ensure that the device driver is initialized before using it. A listing of the device driver, either via SMIT or by using the lsdev command, should indicate the device driver state as Available.

Special File Support: The Multiport Model 2 device driver is a character I/O device and provides a special file entry in the /dev directory for file system access. The Multiport Model 2 device driver uses the t_start_dev and t_chg_parms structures defined in the /usr/include/sys/mpqp.h file to preserve compatibility with the existing GDLC, MPQP API and SNA Services interface. However, only a subset of the #define values is supported for the following t_start_dev structure fields:

- **data_proto**: Identifies the data protocol. The Multiport Model 2 device driver supports the SDLC DATA_PRO_SDLC_HDX value (indicating half duplex only) and the bisync DATA_PRO_BSC value.
- **baud_rate**: Specifies the baud rate for transmit and receive clock. The Multiport Model 2 device driver only supports external clocking where the modem supplies the clock, and this field should be set to zero. However, when using SNA Services, this field is ignored when external clocking is specified in the physical link profile and does not need to be zero.
mpqn Special File

Purpose
Provides access to multiprotocol adapters by way of the Multiprotocol Quad Port (MPQP) device handler.

Description
The /dev/mpqn character special file provides access to the MPQP device handler for the purpose of providing access to a synchronous network. The MPQP device handler supports multiple adapters.

Usage Considerations
When accessing the MPQP device handler, the following should be taken into account:

Driver initialization and termination: The device handler may be loaded and unloaded. The handler supports the configuration calls to initialize and terminate itself.

Special file support: Calls other than the open and close subroutine calls are discussed in relation to the mode in which the device handler is operating.

Subroutine Support
The MPQP device handler supports the open, close, read, write, and ioctl subroutines in the following manner:

- The open and close subroutines
  The device handler supports the /dev/mpqn special file as a character-multiplex special file. The special file must be opened for both reading and writing (O_RDWR). There are no particular considerations for closing the special file. Which special file name is used in an open call depends on how the device is to be opened. Types of special file names are:

  /dev/mpqn Starts the device handler for the selected port.
  /dev/mpqr/D Starts the device handler in Diagnostic mode for the selected port.

- The read subroutine
  Can take the form of a read, readx, readv, or readvx subroutine call. For this call, the device handler copies the data into the buffer specified by the caller.

- The write subroutine
  Can take the form of a write, writex, writev, or writevx subroutine call. For this call, the device handler copies the user data into a buffer and transmits the data on the LAN.

- The ioctl subroutine
  CIO_START Starts a session and registers a network ID.
  CIO_HALT Halts a session and removes a network ID.
  CIO_QUERY Returns the current RAS counter values. These values are defined in the /usr/include/sys/comio.h file.
  CIO_GET_STAT Returns the current adapter and device handler status.
  MP_START_AR Puts the MPQP port into Autoresponse mode.
  MP_STOP_AR Permits the MPQP port to exit Autoresponse mode.
  MP_CHG_PARMS Permits the data link control (DLC) to change certain profile parameters after the MPQP device has been started.
  MP_SET_DELAY Sets the value of NDELAY.

Error Codes
The following error codes may be returned when accessing the device handler through the /dev/mpqn special file:
**ECH RNG** Indicates that the channel number is out of range.

**EAGAIN** Indicates that the maximum number of direct memory accesses (DMAs) was reached, so that the handler cannot get memory for internal control structures.

**EBUSY** Indicates one of the following:
- The port is not in correct state.
- The port should be configured, but is not opened or started.
- The port state is not opened for start of an ioctl operation.
- The port is not started or is not in data-transfer state.

**EIO** Indicates that the handler could not queue the request to the adapter.

**EFAULT** Indicates that the cross-memory copy service failed.

**EINVAL** Indicates one of the following:
- The port not set up properly.
- The handler could not set up structures for write.
- The port is not valid.
- A select operation was called by a kernel process.
- The specified physical-link parameter is not valid for that port.
- The read was called by a kernel process.

**ENOMEM** Indicates one of the following:
- No mbuf or mbuf clusters are available.
- The total data length is more than a page.
- There is no memory for internal structures.

**ENOMSG** Indicates that the status-queue pointer is null, and there are no entries.

**ENOTREADY** Indicates that the port state in define device structure (DDS) is not in Data Transfer mode or that the implicit halt of port failed.

**ENXIO** Indicates one of the following:
- The port has not been started successfully.
- An invalid adapter number was specified.
- The channel number is illegal.
- The adapter is already open in Diagnostic mode.
- The adapter control block (ACB) pointer is null or does not exist.
- The registration of the interrupt handler failed.
- The port does not exist or is not in proper state.
- The adapter number is out of range.

The communication device handler chapter defines specific errors returned on each subroutine call.

**Related Information**

The `close` subroutine, `open` subroutine, `read` or `readx` subroutine, `write` or `writex` subroutine, `ioctl` subroutine.

`MPQP Device Handler Interface Overview` in *AIX 5L Version 5.3 Kernel Extensions and Device Support Programming Concepts*.

**null Special File**

**Purpose**

Provides access to the null device, typically for writing to the bit bucket.
Description
The /dev/null special file provides character access to the null device driver. This device driver is normally accessed to write data to the bit bucket (when the data is to be discarded).

Usage Considerations
When using subroutines with the null device file, consider the following items:

open and close subroutines
The null device can be opened by using the open subroutine with the /dev/null special file name. The close subroutine should be used when access to the null device is no longer required.

read and write subroutines
Data written to this file is discarded. Reading from this file always returns 0 bytes.

ioctl subroutine
There are no ioctl operations available for use with the null special file. Any ioctl operation issued returns with the ENODEV error type.

Related Information
The close subroutine, ioctl subroutine, open subroutine.

nvram Special File

Purpose
Provides access to platform-specific nonvolatile RAM used for system boot, configuration, and fatal error information. This access is achieved through the machine I/O device driver.

Description
The /dev/nvram character special file provides access to the machine device driver for accessing or modifying machine-specific nonvolatile RAM. The appropriate privilege is required to open the nvram special file. The nvram special file is used by machine-specific configuration programs to store or retrieve configuration and boot information using the nonvolatile RAM or ROM provided on the machine. The nvram special file supports open, close, read, and ioctl operations.

Note: Application programs should not access the nonvolatile RAM. Since nonvolatile RAM is platform-specific, any reliance on its presence and implementation places portability constraints upon the using application. In addition, accessing the nonvolatile RAM may cause loss of system startup and configuration information. Such a loss could require system administrative or maintenance task work to rebuild or recover.

For additional information concerning the use of this special file to access machine-specific nonvolatile RAM, see the "Machine Device Driver" in AIX 5L Version 5.3 Technical Reference: Kernel and Subsystems Volume 1.

Usage Considerations
When using subroutines with the nvram special file, consider the following items.

open and close Subroutines: The machine device driver supports the nvram special file as a multiplexed character special file.

A special channel name of base can be used to read the base customize information stored as part of the boot record. The nvram special file must be opened with a channel name of base, as follows:
/dev/nvram/base
The special file `/dev/nvram/base` can only be opened once. When it is closed for the first time after a boot, the buffer containing the base customize information is free. Subsequent opens return a ENOENT error code.

**read, write, and lseek Subroutines:** The `read` subroutine is supported after a successful open of the `nvram` special file with a channel name of `base`. The `read` operation starts transferring data at the location associated with the base customization information and with an offset specified by the offset value associated with the file pointer being used on the subroutine.

On a `read` subroutine, if the end of the data area is reached before the transfer count is reached, the number of bytes read before the end of the data area was reached is returned. If the read starts after the end of the data area, an error of ENXIO is returned by the driver.

The `lseek` subroutine can be used to change the starting read offset within the data area associated with the base customization information. The `write` subroutine is not supported on this channel and results in an error return of ENODEV.

**ioctl Subroutine:** `ioctl` commands can be issued to the machine device driver after the `/dev/nvram` special file has been successfully opened. The IOCINFO parameter returns machine device driver information in the caller's `devinfo` structure, as pointed to by the `arg` parameter to the `ioctl` subroutine. This structure is defined in the `/usr/include/sys/devinfo.h` file. The device type for this device driver is DD_PSEU.

**Error Codes**
The following error conditions can be returned when accessing the machine device driver using the `nvram` special file name:

- **EFAULT** A buffer specified by the caller was invalid on a `read`, `write`, or `ioctl` subroutine call.
- **ENXIO** A read operation was attempted past the end of the data area specified by the channel.
- **ENODEV** A write operation was attempted.
- **ENOMEM** A request was made with a user-supplied buffer that is too small for the requested data.

**Security**
Programs attempting to open the `nvram` special file require the appropriate privilege.

**Files**

`/dev/nvram/base` Allows read access to the base customize information stored as part of the boot record.

**Related Information**

The `close` subroutine, `ioctl` subroutine, `lseek` subroutine, `open` subroutine, `read` subroutine.

**random and urandom Devices**

**Purpose**
Source of secure random output.
Description

The /dev/random and /dev/urandom character devices provide cryptographically secure random output generated from interrupt timings or input written to the devices.

The /dev/random device is intended to provide high quality, cryptographically secure random output and will only return output for which sufficient (an equal or greater amount) random input is available to generate the output. If insufficient random input is available, reads from the /dev/random device will block until the request can be fulfilled unless the O_NONBLOCK flag was specified when the device was opened, in which case as much high quality output as could be generated is returned with the error code EAGAIN.

The /dev/urandom device provides a reliable source of random output, however the output will not be generated from an equal amount of random input if insufficient input is available. Reads from the /dev/urandom device always return the quantity of output requested without blocking. If insufficient random input is available, alternate input will be processed by the random number generator to provide cryptographically secure output, the strength of which will reflect the strength of the algorithms used by the random number generator. Output generated without random input is theoretically less secure than output generated from random input, so /dev/random should be used for applications for which a high level of confidence in the security of the output is required.

Data written to either device is added to the pool of stored random input and may be used for generating output. Writes behave identically for both devices and will not block.

Implementation Specifics

The /dev/random and /dev/urandom devices are created from major and minor numbers assigned by the device configuration subsystem when the random number generator is loaded, so the device names should always be used when attempting to locate or open the devices. The devices are deleted when the random number generator is unloaded. When the system is shut down using the shutdown command, output is taken from the /dev/urandom device and is written back to the /dev/random device when the random number generator is loaded on the next boot to provide starting entropy to the generator, enhancing the quality of the stored random input after boot.

Input is gathered from interrupt timings when the pool of stored random input falls below half full and continues to be gathered until the pool is again full. This process causes a minor performance impact to all external interrupts while timings are being gathered, which ceases when timings cease to be gathered. Data written to either of the random devices will also contribute to the pool of stored random input and can influence the output, thus writing to these devices should be a privileged operation. This is enforced by the permissions of the devices, so it can be changed by the administrator to be completely disallowed if desired.

omd Special File

Purpose

Provides access to the read/write optical device driver.

Description

The omd special file provides block and character (raw) access to disks in the read/write optical drive.

The r prefix on a special file name means that the drive is accessed as a raw device rather than a block device. Performing raw I/O with an optical disk requires that all data transfers be in multiples of the optical-disk logical block length. Also, all lseek subroutines that are made to the raw read/write optical device driver must set the file offset pointer to a value that is a multiple of the specified logical block size.
The scdisk SCSI Device Driver provides more information about implementation specifics.

**Read/Write Optical Device Driver**

Read/write optical disks, used in read/write optical drives, are media that provide storage for large amounts of data. Block access to optical disks is achieved through the special files `/dev/omd0`, `/dev/omd1`, ..., `/dev/omdn`. Character access is provided through the special files `/dev/romd0`, `/dev/romd1`, ..., `/dev/romdn`.

When a read/write optical disk is ejected from the drive for a mounted read/write optical file system, the files on the optical disk can no longer be accessed. Before attempting to access these files again, perform the following steps for a file system mounted from the read/write optical disk:

1. Stop processes that have files open on the file system.
2. Move processes that have current directories on the file system.
3. Unmount the file system.
4. Remount the file system after reinserting the media.

If these actions do not work, perform a forced unmount of the file system; then, remount the file system.

**Note:** Reinserting the read/write optical disk will not fix the problem. Stop all processes (graphical and ASCII), and then forcibly unmount the file system. Then remount the file system. After performing this procedure, you can restart any `man` commands.

**Device-Dependent Subroutines**

Most read/write optical operations are implemented using the `open`, `read`, `write`, and `close` subroutines. However, for some purposes, use of the `openx` (extended) subroutine is required.

**The openx Subroutine**

The `openx` subroutine is supported to provide additional functions to the `open` sequence. Appropriate authority is required for execution. If an attempt is made to run the `openx` subroutine without the proper authority, the subroutine returns a value of -1 and sets the `errno` global variable to a value of EPERM.

**The ioctl Subroutine**

The `ioctl` subroutine `IOCINFO` operation returns the `devinfo` structure defined in the `/usr/include/sys/devinfo.h` file. The `IOCINFO` operation is the only operation defined for all device drivers that use the `ioctl` subroutine. Other `ioctl` operations are specific for the type of device driver. Diagnostic mode is not required for the `IOCINFO` operation.

**Error Conditions**

Possible `errno` values for `ioctl`, `open`, `read`, and `write` subroutines when using the `omd` special file include:

- **EACCES**
  Indicates one of the following circumstances:
  - An attempt was made to open a device currently open in Diagnostic or Exclusive Access mode.
  - An attempt was made to open a Diagnostic mode session on a device already open.
  - The user attempted a subroutine other than an `ioctl` or `close` subroutine while in Diagnostic mode.
  - A `DKIOCMD` operation was attempted on a device not in Diagnostic mode.
  - A `DKFORMAT` operation was attempted on a device not in Exclusive Access mode.

- **EBUSY**
  Indicates one of the following circumstances:
  - The target device is reserved by another initiator.
  - An attempt was made to open a session in Exclusive Access mode on a device already opened.
EFAULT Indicates an illegal user address.
EFORMAT Indicates the target device has unformatted media or media in an incompatible format.
EINVAL Indicates one of the following circumstances:
  • The read or write subroutine supplied an nbyte parameter that is not an even multiple of the block size.
  • A sense data buffer length of greater than 255 bytes is not valid for a DKO IWRSE or DKO IORDSE ioctl subroutine operation.
  • The data buffer length exceeded the maximum defined in the devinfo structure for a DKO IORDSE, DKO IWRSE, or DKO IOCMD ioctl subroutine operation.
  • An unsupported ioctl subroutine operation was attempted.
  • An attempt was made to configure a device that is still open.
  • An illegal configuration command has been given.
  • A DKPMR (Prevent Media Removal), DKAMR (Allow Media Removal), or DKEJECT (Eject Media) command was sent to a device that does not support removable media.
  • A DKEJECT (Eject Media) command was sent to a device that currently has its media locked in the drive.
EIO Indicates one of the following circumstances:
  • The target device cannot be located or is not responding.
  • The target device has indicated an unrecovered hardware error.
EMEDIA Indicates one of the following circumstances:
  • The target device has indicated an unrecovered media error.
  • The media was changed.
EMFILE Indicates an open operation was attempted for an adapter that already has the maximum permissible number of opened devices.
ENODEV Indicates one of the following circumstances:
  • An attempt was made to access an undefined device.
  • An attempt was made to close an undefined device.
ENOTREADY Indicates no read/write optical disk is in the drive.
ENXIO Indicates one of the following circumstances:
  • The ioctl subroutine supplied an invalid parameter.
  • A read or write operation was attempted beyond the end of the physical volume.
EPERM Indicates the attempted subroutine requires appropriate authority.
ESTALE Indicates a read-only optical disk was ejected (without first being closed by the user) and then either reinserted or replaced with a second disk.
ETIMEDOUT Indicates an I/O operation has exceeded the given timer value.
EWRPROTECT Indicates one of the following circumstances:
  • An open operation requesting read/write mode was attempted on read-only media.
  • A write operation was attempted to read-only media.

Files
/dev/romd0, /dev/romd1,..., /dev/romdn Provides character access to the read/write optical device driver.
/dev/omd0, /dev/omd1,..., /dev/omdn Provides block access to the read/write optical device driver.

Related Information
Special Files Overview.
scdisk SCSI Device Driver.
The close subroutine, ioctl subroutine, lseek subroutine, open subroutine, read subroutine, write subroutine.

**Open Special File**

**Purpose**
Provides a diagnostic interface to the serial optical ports by way of the Serial Optical Link device driver.

**Description**
The opn character special file provides strictly diagnostic access to a specific serial optical port. The normal interface to the serial optical link is through the ops0 special file.

**Related Information**

**Opn Special File**

**Purpose**
Provides access to the serial optical link by way of the Serial Optical Link Device Handler Overview.

**Description**
The Serial Optical Link device driver is a component of the Communication I/O subsystem. The device driver can support from one to four serial optical ports. An optical port consists of two separate pieces. The Serial Link Adapter is on the system planar, and is packaged with two to four adapters in a single chip. The Serial Optical Channel Converter plugs into a slot on the system planar and provides two separate optical ports.

The ops0 special file provides access to the optical port subsystem. An application that opens this special file has access to all the ports, but does not need to be aware of the number of ports available. Each write operation will include a destination processor ID, and the device driver will route the data through the correct port to reach that processor. If there is more than one path to the destination, the device driver will use any link that is available, in case of a link failure.

**Usage Considerations**
When accessing the Serial Optical Link device driver, the following should be taken into account:

**Driver initialization and termination**
The device driver may be loaded and unloaded. The device driver supports the configuration calls to initialize and terminate itself.

**Special file support**
Calls other than the open and close subroutines are discussed based on the mode in which the device driver is operating.

**Subroutine Support**
The Serial Optical Link device driver provides specific support for the open, close, read, write, and ioctl subroutines.

**Open and close Subroutines:** The device driver supports the /dev/ops0 special file as a character-multiplex special file. The special file must be opened for both reading and writing (O_RDWR). There are no particular considerations for closing the special file. The special file name is used in an open
call depending on how the device is to be opened. The two types of special file names are:

/dev/ops0  Starts the device driver in normal mode.
/dev/ops0/S Starts the device driver in serialized mode. As a result, the device driver transmits data in the same order in which it receives the data.

**read Subroutine:** Can take the form of a `read`, `readx`, `readv`, or `readvx` subroutine. For this call, the device driver copies the data into the buffer specified by the caller.

**write Subroutine:** Can take the form of a `write`, `writex`, `writev`, or `writevx` subroutine. For this call, the device driver copies the user data into a kernel buffer and transmits the data.

**ioctl subroutine:** The Serial Optical Link device driver supports the following `ioctl` operations:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIO_GET_FASTWRT</td>
<td>Gets attributes needed for the sol_fastwrt entry point.</td>
</tr>
<tr>
<td>CIO_GET_STAT</td>
<td>Gets device status.</td>
</tr>
<tr>
<td>CIO_HALT</td>
<td>Halts the device.</td>
</tr>
<tr>
<td>CIO_QUERY</td>
<td>Queries device statistics.</td>
</tr>
<tr>
<td>CIO_START</td>
<td>Starts the device.</td>
</tr>
<tr>
<td>IOCINFO</td>
<td>I/O character information.</td>
</tr>
<tr>
<td>SOL_CHECK_PRID</td>
<td>Checks if a processor ID is connected.</td>
</tr>
<tr>
<td>SOL_GET_PRIDS</td>
<td>Gets connected processor IDs.</td>
</tr>
</tbody>
</table>

**Error Codes**
The following error codes may be returned when accessing the device driver through the `/dev/ops0` special file:

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EACCES</td>
<td>Indicates access to the device is denied for one of the following reasons:</td>
</tr>
<tr>
<td></td>
<td>• A non-privileged user tried to open the device in Diagnostic mode.</td>
</tr>
<tr>
<td></td>
<td>• A kernel-mode user attempted a user-mode call.</td>
</tr>
<tr>
<td></td>
<td>• A user-mode user attempted a kernel-mode call.</td>
</tr>
<tr>
<td>EADDRINUSE</td>
<td>Indicates the network ID is in use.</td>
</tr>
<tr>
<td>EAGAIN</td>
<td>Indicates that the transmit queue is full.</td>
</tr>
<tr>
<td>EBUSY</td>
<td>Indicates one of the following:</td>
</tr>
<tr>
<td></td>
<td>• The device was already initialized.</td>
</tr>
<tr>
<td></td>
<td>• There are outstanding opens; unable to terminate.</td>
</tr>
<tr>
<td></td>
<td>• The device is already open in Diagnostic mode.</td>
</tr>
<tr>
<td></td>
<td>• The maximum number of opens has been exceeded.</td>
</tr>
<tr>
<td>EFAULT</td>
<td>Indicates that the specified address is not valid.</td>
</tr>
<tr>
<td>EINTR</td>
<td>Indicates that a system call was interrupted.</td>
</tr>
<tr>
<td>EINVAL</td>
<td>Indicates that the specified parameter is not valid.</td>
</tr>
<tr>
<td>EIO</td>
<td>Indicates a general error. If an extension was provided in the call,</td>
</tr>
<tr>
<td></td>
<td>additional data identifying the cause of the error can be found in the</td>
</tr>
<tr>
<td></td>
<td>status field.</td>
</tr>
<tr>
<td>EMSGSIZE</td>
<td>Indicates that the data was too large to fit into the receive buffer and</td>
</tr>
<tr>
<td></td>
<td>that no <code>arg</code> parameter was supplied to provide an alternate means of</td>
</tr>
<tr>
<td></td>
<td>reporting this error with a status of <code>CIO_BUF_OVFLW</code>.</td>
</tr>
<tr>
<td>ENETDOWN</td>
<td>Indicates that the network is down. The device is unable to process the</td>
</tr>
<tr>
<td>ENOCONNECT</td>
<td>Indicates one of the following:</td>
</tr>
<tr>
<td></td>
<td>• The device is not started.</td>
</tr>
<tr>
<td></td>
<td>• The processor ID is not connected to the Serial Optical Link subsystem.</td>
</tr>
<tr>
<td>ENODEV</td>
<td>Indicates that the specified minor number is not valid.</td>
</tr>
<tr>
<td>ENOMEM</td>
<td>Indicates that the device driver was unable to allocate the required</td>
</tr>
<tr>
<td>ENOSPC</td>
<td>Indicates the network ID table is full.</td>
</tr>
<tr>
<td>EPERM</td>
<td>Indicates that the device is open in a mode that does not allow a</td>
</tr>
</tbody>
</table>

AIX 5L Version 5.3 Files Reference
Related Information
The `close` subroutine, `open` subroutine, `read` or `readx` subroutine, `write` or `writex` subroutine, `ioctl` subroutine.


pty Special File

Purpose
Provides the pseudo-terminal (pty) device driver.

Description
The pty device driver provides support for a *pseudo-terminal*. A pseudo-terminal includes a pair of control and slave character devices. The slave device provides processes with essentially the same interface as that provided by the `tty` device driver. However, instead of providing support for a hardware device, the slave device is manipulated by another process through the control half of the pseudo-terminal. That is, anything written on the control device is given to the slave device as input and anything written on the slave device is presented as input on the control device.

In AIX Version 4, the pty subsystem uses naming conventions similar to those from UNIX System V. There is one node for the control driver, `/dev/ptc`, and a maximum number of `N` nodes for the slave drivers, `/dev/pts/n`. `N` is configurable at pty configuration and may be changed dynamically by pty reconfiguration, without closing the opened devices.

The control device is set up as a *clone device* whose major device number is the clone device’s major number and whose minor device number is the control driver’s major number. There is no node in the filesystem for control devices. A control device can be opened only once, but slave devices can be opened several times.

By opening the control device with the `/dev/ptc` special file, an application can quickly open the control and slave sides of an unused pseudo-terminal. The name of the corresponding slave side can be retrieved using the `ttyname` subroutine, which always returns the name of the slave side.

With Berkeley pty subsystems, commands have to search for an unused pseudo-terminal by opening each control side sequentially. The control side could not be opened if it was already in use. Thus, the opens would fail, setting the `errno` variable to `EIO`, until an unused pseudo-terminal was found. It is possible to configure the pty subsystem in order to use special files with the BSD pty naming convention:

```
Control devices /dev/pty[p-zA-Z][0-f]
Slave devices /dev/tty[p-zA-Z][0-f]
```

These special files are not symbolic links to the operating system special files. They are completely separate. The number of control and slave pair devices using the BSD naming convention is configurable.

The following ioctl commands apply to pseudo-terminals:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIOCSTOP</td>
<td>Stops output to a terminal. This is the same as using the Ctrl-S key sequence. No parameters are allowed for this command.</td>
</tr>
<tr>
<td>TIOCSTART</td>
<td>Restarts output that was stopped by a TIOCSTOP command or by the Ctrl-S key sequence. This is the same as typing the Ctrl-Q key sequence. No parameters are allowed for this command.</td>
</tr>
</tbody>
</table>
TIOCPKT

Enables and disables the packet mode. Packet mode is enabled by specifying (by reference) a nonzero parameter. It is disabled by specifying (by reference) a zero parameter. When applied to the control side of a pseudo-terminal, each subsequent read from the terminal returns data written on the slave part of the pseudo terminal. The data is preceded either by a zero byte (symbolically defined as TIOCPKT_DATA) or by a single byte that reflects control-status information. In the latter case, the byte is an inclusive OR of zero or more of the following bits:

TIOCPKT_FLUSHREAD
The read queue for the terminal is flushed.

TIOCPKT_FLUSHWRITE
The write queue for the terminal is flushed.

TIOCPKT_STOP
Output to the terminal is stopped with Ctrl-S.

TIOCPKT_START
Output to the terminal is restarted.

TIOCPKT_DOSTOP
The stop character defined by the current tty line discipline is Ctrl-S; the start character defined by the line discipline is Ctrl-Q.

TIOCPKT_NOSTOP
The start and stop characters are not Ctrl-S and Ctrl-Q.

While this mode is in use, the presence of control-status information to be read from the control side can be detected by a select for exceptional conditions.

This mode is used by the rlogin and rlogind commands to log in to a remote host and implement remote echoing and local Ctrl-S and Ctrl-Q flow control with proper back-flushing of output.

TIOCUCNTRL

Enables and disables a mode that allows a small number of simple user ioctl commands to be passed through the pseudo-terminal, using a protocol similar to that of the TIOCPKT mode. The TIOCUCNTRL and TIOCPKT modes are mutually exclusive.

This mode is enabled from the control side of a pseudo-terminal by specifying (by reference) a nonzero parameter. It is disabled by specifying (by reference) a zero parameter. Each subsequent read from the control side will return data written on the slave part of the pseudo-terminal, preceded either by a zero byte or by a single byte that reflects a user control operation on the slave side.

A user-control command consists of a special ioctl operation with no data. That command is issued as UIOCCMD(Value), where the Value parameter specifies a number in the range 1 through 255. The operation value is received as a single byte on the next read from the control side.

A value of 0 can be used with the UIOCCMD ioctl operation to probe for the existence of this facility. The zero is not made available for reading by the control side. Command operations can be detected with a select for exceptional conditions.

TIOCREMOTE

A mode for the control half of a pseudo-terminal, independent of TIOCPKT. This mode implements flow control, rather than input editing, for input to the pseudo-terminal, regardless of the terminal mode. Each write to the control terminal produces a record boundary for the process reading the terminal. In normal usage, a write of data is like the data typed as a line on the terminal, while a write of zero bytes is like typing an end-of-file character. This mode is used for remote line editing in a window-manager and flow-controlled input.

Related Information
The rlogin command, rlogind command.

The ioctl subroutine, ttymame subroutine.
rcm Special File

Purpose
Provides the application interface to obtain and relinquish the status of a graphics process through the Rendering Context Manager (RCM) device driver.

Description
The rcm is used by graphics systems to obtain a gsc_handle. This handle is required in the call to aixgsc which is part of the procedure of becoming a graphics process.

Usage Considerations
The RCM device driver supports open, close, and ioctl subroutines.

A application uses the GSC_HANDLE ioctl command to get a gsc_handle as part of becoming a graphics process. When it closes rcm, either normally, or by abnormal termination, the RCM releases any displays which it owns. This is implemented as a LFT_REL_DISP ioctl command to the LFT device driver.

IOCINFO Returns devinfo structure.
GSC_HANDLE Returns a gsc_handle.
RCM_SET_DIAG_OWNER Obtain exclusive use of the display adapter for diagnostics.

Related Information
lft Special File.
Special Files Overview.

rhdisk Special File

Purpose
Provides raw I/O access to the physical volumes (fixed-disk) device driver.

Description
The rhdisk special file provides raw I/O access and control functions to physical-disk device drivers for physical disks. Raw I/O access is provided through the /dev/rhdisk0, /dev/rhdisk1, ..., character special files.

Direct access to physical disks through block special files should be avoided. Such access can impair performance and also cause data consistency problems between data in the block I/O buffer cache and data in system pages. The /dev/hdisk block special files are reserved for system use in managing file systems, paging devices and logical volumes.

The r prefix on the special file name indicates that the drive is to be accessed as a raw device rather than a block device. Performing raw I/O with a fixed disk requires that all data transfers be in multiples of the disk block size. Also, all lseek subroutines that are made to the raw disk device driver must result in a file-pointer value that is a multiple of the disk-block size.
Usage Considerations

Note: Data corruption, loss of data, or loss of system integrity (system crashes) will occur if devices supporting paging, logical volumes, or mounted file systems are accessed using block special files. Block special files are provided for logical volumes and disk devices on the operating system and are solely for system use in managing file systems, paging devices, and logical volumes. They should not be used for other purposes.

**open and close Subroutines:** The `open` subroutine provides additional functions to the open sequence. This subroutine requires appropriate permission to execute. Attempting to do so without the proper permission results in a return value of -1, with the `errno` global variable set to **EPERM**.

**read and write Subroutines:** The `read` and `write` subroutines provide for additional parameters affecting the raw data transfer. The `ext` parameter specifies certain options that apply to the request being made. The options are constructed by logically ORing zero or more of the following values.

Note: The following operations can be used only with the `write` subroutine.

- WRITEV: Perform physical write verification on this request.
- HWRELOC: Perform hardware relocation of the specified block before the block is written. This is done only if the drive supports safe relocation. Safe relocation ensures that once the relocation is started, it will complete safely regardless of power outages.
- UNSAFEREL: Perform hardware relocation of the specified block before the block is written. This is done if the drive supports any kind of relocation (safe or unsafe).

**ioctl Subroutine:** Only one `ioctl` operation, `IOCINFO`, is defined for all device drivers that use the `ioctl` subroutine. The remaining `ioctl` operations are all specific to physical-disk devices. Diagnostic mode is not required for the `IOCINFO` operation.

The `IOCINFO` ioctl operation returns a structure for a device type of **DD_DISK**. This structure is defined in the `/usr/include/sys/devinfo.h` file.

**Error Codes**

In addition to the errors listed for the `ioctl`, `open`, `read`, and `write` subroutines, the following other error codes are also possible:

- **EACCESS** An `open` subroutine call has been made to a device in Diagnostic mode.
- **EACCESS** A diagnostic `open` subroutine call has been made to a device already opened.
- **EACCESS** A diagnostic `ioctl` operation has been attempted when not in Diagnostic mode.
- **EINVAL** An `nbyte` parameter to a `read` or `write` subroutine is not a multiple of the disk block size.
- **EINVAL** An unsupported `ioctl` operation has been attempted.
- **EINVAL** An unsupported `read` or `write` subroutine has been attempted.
- **EMEDIA** The target device has indicated an unrecovered media error.
- **ENXIO** A parameter to the `ioctl` subroutine is invalid.
- **ENXIO** A `read` or `write` subroutine has been attempted beyond the end of the disk.
- **EIO** The target device cannot be located or is not responding.
- **EIO** The target device has indicated an unrecovered hardware error.
- **EMFILE** An `open` subroutine has been attempted for an adapter that already has the maximum permissible number of opened devices.
- **EPERM** The caller lacks the appropriate privilege.

**Files**

`/dev/hdisk0, /dev/hdisk1, ... /dev/hdiskn` Provide block I/O access to the physical volumes (fixed-disk) device driver.
Related Information

The close subroutine, ioctl subroutine, lseek subroutine, open subroutine, read subroutine, write subroutine.


rmt Special File

Purpose
Provides access to the sequential-access bulk storage medium device driver.

Description
Magnetic tapes are used primarily for backup, file archives, and other off-line storage. Tapes are accessed through the /dev/rmt*, ... , /dev/rmt255 special files. The r in the special file name indicates raw access through the character special file interface. A tape device does not lend itself well to the category of a block device. Thus, only character interface special files are provided.

Special files associated with each tape device determine which action is taken during open or close operations. These files also dictate, for applicable devices, at what density data is to be written to tape. The following table shows the names of these special files and their corresponding characteristics:

<table>
<thead>
<tr>
<th>Special File Name</th>
<th>Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>/dev/rmt*</td>
<td>Rewind-on-Close  Yes</td>
</tr>
<tr>
<td></td>
<td>Retension-on-Open No</td>
</tr>
<tr>
<td></td>
<td>Bytes per Inch   Density setting #1</td>
</tr>
<tr>
<td>/dev/rmt*.1</td>
<td>Rewind-on-Close  No</td>
</tr>
<tr>
<td></td>
<td>Retension-on-Open No</td>
</tr>
<tr>
<td></td>
<td>Bytes per Inch   Density setting #1</td>
</tr>
<tr>
<td>/dev/rmt*.2</td>
<td>Rewind-on-Close  Yes</td>
</tr>
<tr>
<td></td>
<td>Retension-on-Open Yes</td>
</tr>
<tr>
<td></td>
<td>Bytes per Inch   Density setting #1</td>
</tr>
<tr>
<td>Special File Name</td>
<td>Characteristics</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------------------------------</td>
</tr>
</tbody>
</table>
| /dev/rmt*.3      | Rewind-on-Close  
|                  |    No                                     |
|                  | Retension-on-Open  
|                  |    Yes                                   |
|                  | Bytes per Inch  
|                  |    Density setting #1                     |
| /dev/rmt*.4      | Rewind-on-Close  
|                  |    Yes                                   |
|                  | Retension-on-Open  
|                  |    No                                    |
|                  | Bytes per Inch  
|                  |    Density setting #2                     |
| /dev/rmt*.5      | Rewind-on-Close  
|                  |    No                                    |
|                  | Retension-on-Open  
|                  |    No                                    |
|                  | Bytes per Inch  
|                  |    Density setting #2                     |
| /dev/rmt*.6      | Rewind-on-Close  
|                  |    Yes                                   |
|                  | Retension-on-Open  
|                  |    Yes                                   |
|                  | Bytes per Inch  
|                  |    Density setting #2                     |
| /dev/rmt*.7      | Rewind-on-Close  
|                  |    No                                    |
|                  | Retension-on-Open  
|                  |    Yes                                   |
|                  | Bytes per Inch  
|                  |    Density setting #2                     |

1. The values of density setting #1 and density setting #2 come from tape drive attributes that can be set using SMIT. Typically density setting #1 is set to the highest possible density for the tape drive while density setting #2 is set to a lower density. However, density settings are not required to follow this pattern.

2. The density value (bytes per inch) is ignored when using a magnetic tape device that does not support multiple densities. For tape drives that do support multiple densities, the density value only applies when writing to the tape. When reading, the drive defaults to the density at which the tape is written.

3. Most tape drives use 512-byte block size. The 8mm tape drive uses a minimum block size of 1024 bytes. Using SMIT to lower the block size, will waste space.

**Usage Considerations**

Most tape operations are implemented using the `open`, `read`, `write`, and `close` subroutines. However, for diagnostic purposes, the `openx` subroutine is required.
open and close Subroutines:  Care should be taken when closing a file after writing. If the application reverses over the data just written, no file marks will be written. However, for tape devices that allow for block update, unless the application spaces in the reverse direction or returns the tape position to the beginning of tape (BOT), one or two file marks will be written upon closing the device. (The number of file marks depends on the special file type.)

For multitape jobs, the special file must be opened and closed for each tape. The user is not allowed to continue if the special file is opened and the tape has been changed.

The openx subroutine is intended primarily for use by the diagnostic commands and utilities. Appropriate authority is required for execution. Executing this subroutine without the proper authority results in a return value of -1, with the errno global variable set to EPERM.

read and write Subroutines:  When opened for reading or writing, the tape is assumed to be positioned as desired. When the tape is opened as no-rewind-on-close (/dev/rmt*.1) and a file is written, a single file mark is written upon closing the tape. When the tape is opened as rewind-on-close (/dev/rmt*) and a file is written, a double file mark is written upon closing the tape. When the tape is opened as no-rewind-on-close and reads from a file, the tape is positioned upon closing after the end-of-file (EOF) mark following the data just read.

By specifically choosing the rmt file, it is possible to create multiple file tapes.

Although tapes are accessed through character interface special files, the number of bytes required by either a read or write operation must be a multiple of the block size defined for the magnetic tape device. When the tape drive is in variable block mode, read requests for less than the tape’s block size return the number of bytes requested and set the errno global variable to a value of 0. In this case, the readx subroutine’s Extension parameter must be set to TAPE_SHORT_READ.

During a read, the record size is returned as the number of bytes read, up to the buffer size specified. If an EOF condition is encountered, then a zero-length read is returned, with the tape positioned after the EOF.

An end-of-media (EOM) condition encountered during a read or write operation results in the return of the number of bytes successfully ready or written. When a write is attempted after the device has reached the EOM, a value of -1 is returned with the errno global variable set to the ENXIO value. When a read is attempted after the device has reached the EOM, a zero-length read is returned. Successive reads continue to return a zero-length read.

Data Buffering With a Tape Device:  Some tape devices contain a data buffer to maximize data transfer speed when writing to tape. A write operation sent to tape is returned as complete when the data is transferred to the data buffer of the tape device. The data in the buffer is then written to tape asynchronously. As a result, data-transfer speed increases since the host need not wait for I/O completion.

Two modes are provided by the tape device driver to facilitate use of these data buffers. The non-buffered mode causes writes to tape to bypass the data buffer and go directly to tape. In buffered mode, all write subroutines are returned as complete when the transfer data has been successfully written to the tape device buffer. The device driver does not flush the data buffer until the special file is closed or an EOM condition is encountered.

If an EOM condition is encountered while running in buffered mode, the device attempts to flush the device data buffer. The residual count can exceed the write transfer length in buffered mode. In some cases, the flushing of residual data may actually run the tape off the reel. Either case is considered a failure and results in a return value of -1, with the errno global variable set to EIO. These errors can require the user to run in non-buffered mode.
**rmt Special File Considerations:** Failures that result in a device reset while reading or writing to tape require the special file to be closed and the job restarted. Any commands issued after this condition occurs and until the special file is closed result in a return value of -1, with the `errno` global variable set to `EIO`. Non-reset type errors (that is, media or hardware errors) result in the tape being left positioned where the error occurred.

For multi-tape jobs, the special file must be opened and closed for each tape. The user is not allowed to continue if the special file is opened and the tape has been changed.

A signal received by the tape device driver will cause the current command to abort. As a result, the application halts time-consuming commands (for instance, an erase operation) without recycling the drive power or waiting for a timeout to occur.

Use of zero (0) as a block-size parameter indicates the blocksize is of variable length.

**ioctl Subroutine:** A single `ioctl` operation, `IOCINFO`, is defined for all device drivers that use the `ioctl` subroutine. For the `rmt` special file, the `STIOCTOP` operation has also been defined.

The **IOCINFO ioctl operation:** The `IOCINFO` ioctl operation returns a structure defined in the `/usr/include/sys/devinfo.h` file.

The **STIOCTOP ioctl operation:** The `STIOCTOP` ioctl operation provides for command execution operations, such as erase and retension. The parameter to the `ioctl` subroutine using the `STIOCTOP` operation specifies the address of a `stop` structure, as defined in the `/usr/include/sys/tape.h` file.

The operation found in the `st_op` field in the `stop` structure is performed `st_count` times, except for rewind, erase, and retension operations. The available operations are:

- **STREW** Rewind.
- **STOFFL** Rewind and unload the tape. A tape must be inserted before the device can be used again.
- **STERASE** Erase tape; leave at load point.
- **STRETEN** Retension tape; leave at load point.
- **STWEOF** Write and end-of-file mark.
- **STFSF** Forward space file.
- **STFSR** Forward space record.
- **STRSF** Reverse space file.
- **STRSR** Reverse space record.
- **STDEOF** Disable end-of-file check.

**Note:** Use of the `STDEOF` command enables an application to write beyond the end of the tape. When disabling end-of-file checking by issuing the `STDEOF` command, it is the responsibility of the application to guard against error conditions that can arise from the use of this command.

**Note:** Execution of the preceding commands depends on the particular tape device and which commands are supported. If the command is not supported on a particular device, a value of -1 is returned, with the `errno` global variable set to `EINVAL`.

**Error Codes**
In addition to general error codes listed for `ioctl`, `open`, `read`, and `write` subroutines, the following specific error codes may also occur:

- **EAGAIN** An open operation was attempted to a device that is already open.
- **EBUSY** The target device is reserved by another initiator.
- **EINVAL** An `nbyte` parameter to a `read` or `write` subroutine is not an even multiple of the blocksize.
- **EINVAL** A parameter to the `ioctl` subroutine is invalid.
- **EINVAL** The requested `ioctl` operation is not supported on the current device.
EIO          Could not space forward or reverse st_count records before encountering an EOM condition or a file mark.
EIO          Could not space forward or reverse st_count file marks before encountering an EOM condition.
EMEDIA       The tape device has encountered an unrecoverable media error.
ENOMEM       The number of bytes requested for a read of a variable-length record on tape is less than the actual size (in bytes) of the variable-length record.
ENOTREADY    There is no tape in the drive or the drive is not ready.
ENXIO        A write operation was attempted while the tape was at the EOM.
EPERM        The requested subroutine requires appropriate authority.
ETIMEDOUT    A command has timed out.
EWRPROTECT   An open operation for read/write was attempted on a read-only tape.
EWRPROTECT   An ioctl operation that effects media was attempted on a read-only tape.

Related Information

The close subroutine, ioctl subroutine, open subroutine, openx subroutine, read subroutine, write subroutine.

scsi Special File

Purpose
Provides access to the SCSI adapter driver.

Description
The scsi special file provides an interface to an attached SCSI adapter. This special file should not be opened directly by application programs (with the exception of diagnostics applications). The /dev/scsi0, /dev/scsi1, ... /dev/scsiN files are the scsi special files.

The description of the SCSI Adapter device driver in AIX 5L Version 5.3 Technical Reference: Kernel and Subsystems Volume 1 provides the implementation specifics for the SCSI adapter.

Related Information


The scdisk SCSI Device Driver, and rmt SCSI Device Driver.

tablet Special File

Purpose
Provides access to the tablet.

Description
The tablet special file is the application interface to the tablet. It provides the applications with the capability of receiving input from the tablet and it allows the application to change the sampling rate, dead zones, origin, resolution, and conversion mode.
Configuration
There are no user commands to change the configuration of the tablet device. Applications may use `ioctl` commands to modify the configuration but these modifications are effective only until the tablet is closed.

Usage Considerations
The `open` subroutine call specifying the `tablet` special file is processed normally except that the `Oflag` and `Mode` parameters are ignored. The open request is rejected if the special file is already opened or if a kernel extension attempts to open the special file. All tablet inputs are flushed following an `open` subroutine call until an input ring is established. The tablet device is reset to the default configuration when an open request is made.

The `tablet` special file does not support the `read` or `write` subroutine calls. Instead, input data is obtained from the tablet through the input ring. The `read` and `write` subroutine calls behave the same as `read` or `write` subroutine calls to the `/dev/null` file.

The `tablet` special file supports the following functions with `ioctl` subroutines:

- **IOCINFO**: Returns `devinfo` structure.
- **TABCONVERSION**: Sets tablet conversion mode.
- **TABDEADZONE**: Sets tablet dead zones.
- **TABFLUSH**: Flushes input ring.
- **TABORIGIN**: Sets tablet origin.
- **TABQUERYID**: Queries tablet device identifier.
- **TABREGRING**: Registers input ring.
- **TABRESOLUTION**: Sets resolution.
- **TABSAMPELRATE**: Sets sample rate.

Error Codes
The error codes can be found in the `/usr/include/sys/errno.h` file.

- **EFAULT**: Indicates insufficient authority to access address or invalid address.
- **EIO**: Indicates an I/O error.
- **ENOMEM**: Indicates insufficient memory for required paging operation.
- **ENOSPC**: Indicates insufficient file system or paging space.
- **EINVAL**: Indicates an invalid argument.
- **EINTR**: Indicates the request was interrupted by signal.
- **EPERM**: Indicates a permanent error occurred.
- **EBUSY**: Indicates the device is busy.
- **ENXIO**: Indicates an unsupported device number was specified.
- **ENODEV**: Indicates an unsupported device or device type mismatch.
- **EACCES**: Indicates `open` is not allowed.

Files
`/usr/include/sys/inputdd.h` contains declarations for `ioctl` commands and input ring report format.

Related Information
The `close` subroutine, `ioctl` subroutine, `open` subroutine, `read` subroutine, `write` subroutine.
The **dial** special file, **GIO** special file, **kbd** special file, **lpf** special file, **mouse** special file.

**Special Files Overview**

---

**tmscsi Special File**

**Purpose**

Provides access to the SCSI **tmscsi** device driver.

**Description**

The **tmscsi** special file provides an interface to allow processor-to-processor data transfer using the SCSI **send** command. This single device driver handles both SCSI initiator and SCSI target mode roles.

The user accesses the data transfer functions through the special files `/dev/tmscsi0.xx`, `/dev/tmscsi1.xx`, .... These are all character special files. The `xx` variable can be either **im**, initiator-mode interface, or **tm**, target-mode interface. The initiator-mode interface transmits data, and the target-mode interface receives data.

The least significant bit of the minor device number indicates to the device driver which mode interface is selected by the caller. When the least significant bit of the minor device number is set to 1, the target-mode interface is selected. When the least significant bit is set to 0, the initiator-mode interface is selected.

When the caller opens the initiator-mode special file, a logical path is established allowing data to be transmitted. The **write**, **writex**, **writev**, or **writevx** subroutine initiates data transmission for a user-mode caller, and the **fp_write** or **fp_rwuio** kernel services initiate data transmission for a kernel-mode caller. The SCSI target-mode device driver then builds a SCSI **send** command to describe the transfer, and the data is sent to the device. Once the write entry point returns, the calling program can access the transmit buffer.

When the caller opens the target-mode special file, a logical path is established allowing data to be received. The **read**, **readx**, **readv**, or **readvx** subroutine initiates data reception for a user-mode caller, and the **fp_read** or **fp_rwuio** kernel service initiates data reception for a kernel-mode caller. The SCSI target-mode device driver then returns data received for the application.

**Note:** This operation is not supported by all SCSI I/O controllers.

**Related Information**

**tmscsi**

The **close** subroutine, **open** subroutine, **read** or **readx** subroutine, **write** or **writex** subroutine.

---

**tokn Special File**

**Purpose**

Provides access to the token-ring adapters by way of the token-ring device handler.
Description
The `tokn` character special file provides access to the token-ring device handler that provides access to a
token-ring local area network. The device handler supports up to four token-ring adapters.

Usage Considerations
When accessing the token-ring device handler, the following should be taken into account:

**Driver initialization and termination:** The device handler may be loaded and unloaded. The device
handler supports the configuration calls to initialize and terminate itself.

**Special file support:** Calls other than the `open` and `close` subroutines are discussed based on the
mode in which the device handler is operating.

Subroutine Support
The token-ring device handler provides specific support for the `open`, `close`, `read`, `write`, and `ioctl`
subroutines.

**open and close Subroutines:** The device handler supports the `/dev/tokn` special file as a
character-multiplex special file. The special file must be opened for both reading and writing (`O_RDWR`).
There are no particular considerations for closing the special file. The special file name used in an `open`
call depends upon how the device is to be opened. The three types of special file names are:

```
/dev/tokn  Starts the device handler for the selected port, where the value of `n` is 0 <= `n` <= 7.
/dev/tokn/D  Starts the device handler for the selected port in Diagnostic mode, where the value of `n`
is 0 <= `n` <= 7.
/dev/tokn/W  Starts the device handler for the selected port in Diagnostic Wrap mode, where the value of `n`
is 0 <= `n` <= 7.
```

**read Subroutine:** Can take the form of a `read`, `readx`, `readv`, or `readvx` subroutine. For this call, the
device handler copies the data into the buffer specified by the caller.

**write Subroutine:** Can take the form of a `write`, `writex`, `writev`, or `writevx` subroutine. For this call, the
device handler copies the user data into a kernel buffer and transmits the data on the LAN.

**ioctl Subroutine:** The token-ring device handler supports the following `ioctl` operations:

<table>
<thead>
<tr>
<th><code>ioctl</code></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CIO_GET_STAT</code></td>
<td>Returns current adapter and device handler status.</td>
</tr>
<tr>
<td><code>CIO_HALT</code></td>
<td>Halts a session and removes a network ID from the network ID table.</td>
</tr>
<tr>
<td><code>CIO_QUERY</code></td>
<td>Returns the current counter values, as defined in the <code>/usr/include/sys/comio.h</code> and <code>/usr/include/sys/tokuser.h</code> files.</td>
</tr>
<tr>
<td><code>CIO_START</code></td>
<td>Starts a session and registers a network ID.</td>
</tr>
<tr>
<td><code>IOCINFO</code></td>
<td>Returns a structure of device information to the user specified area. The <code>devtype</code> field is <code>DD_NET_DH</code> and the <code>devsubtype</code> field is <code>DD_TR</code>, as defined in the <code>/usr/include/sys/devinfo.h</code> file.</td>
</tr>
<tr>
<td><code>TOK_GRP_ADDR</code></td>
<td>Allows the setting of the active group address for the token-ring adapter.</td>
</tr>
<tr>
<td><code>TOK_FUNC_ADDR</code></td>
<td>Allows the setting of a functional address for the token-ring adapter.</td>
</tr>
<tr>
<td><code>TOK_QVPD</code></td>
<td>Returns adapter vital product data.</td>
</tr>
<tr>
<td><code>TOK_RING_INFO</code></td>
<td>Returns information about the token-ring device.</td>
</tr>
</tbody>
</table>

Error Conditions
The following error conditions may be returned when accessing the device handler through the `dev/tokn`
special file:
EACCES Indicates that permission to access the adapter is denied for one of the following reasons:
- Device has not been configured.
- Diagnostic mode open request denied.
- The call is from a kernel-mode process.

EAGAIN Indicates that the transmit queue is full.

EBUSY Indicates one of the following:
- The device is already opened in Diagnostic mode.
- The maximum number of opens has already been reached.
- The request is denied.
- The device is in use.
- The device handler cannot terminate.

EEXIST Indicates that the device is already configured or the device handler is unable to remove the device from switch table.

EFAULT Indicates that an invalid address or parameter was specified.

EINVAL Indicates one of the following:
- The parameters specified were invalid.
- The define device structure (DDS) is invalid.
- The device handler is not in Diagnostic mode.

ENOCONNECT Indicates that the device has not been started.

ENETDOWN Indicates that the network is down and the device handler is unable to process the command.

ENOENT Indicates that there was no DDS available.

ENOMEM Indicates that the device handler was unable to allocate required memory.

ENOMSG Indicates that there was no message of desired type.

ENOSPC Indicates that the network ID table is full or the maximum number of opens was exceeded.

EADDRINUSE Indicates that the specified network ID is in use.

ENXIO Indicates that the specified minor number was not valid.

ENETUNREACH Indicates that the device handler is in Network Recovery mode and is unable to process the write operation.

EMSGSIZE Indicates that the data is too large for the supplied buffer.

Related Information
The close subroutine, open subroutine, read or readx subroutine, write or writex subroutine.

---

trace Special File

Purpose
Supports event tracing.

Description
The /dev/systrace and /dev/systrcctl special files support the monitoring and recording of selected system events. Minor device 0 of the trace drivers is the interface between processes that record trace events and the trace daemon. Write trace events to the /dev/systrace file by the trchk and trcgen subroutines and the trcgenk kernel service. Minor devices 1 through 7 of the trace drivers support generic trace channels for tracing system activities such as communications link activities.

The trace special file is part of Software Trace Service Aids package.

Related Information
The trcgenk kernel service.
tty Special File

Purpose
Supports the controlling terminal interface.

Description
For each process, the /dev/tty special file is a synonym for the controlling terminal associated with that process. By directing messages to the tty file, application programs and shell sequences can ensure that the messages are written to the terminal even if output is redirected. Programs can also direct their display output to this file so that it is not necessary to identify the active terminal.

A terminal can belong to a process as its controlling terminal. Each process of a session that has a controlling terminal has the same controlling terminal. A terminal can be the controlling terminal for one session at most. If a session leader has no controlling terminal and opens a terminal device file that is not already associated with a session (without using the O_NOCTTY option of the open subroutine), the terminal becomes the controlling terminal of the session leader. If a process that is not a session leader opens a terminal file or if the O_NOCTTY option is used, that terminal does not become the controlling terminal of the calling process. When a controlling terminal becomes associated with a session, its foreground process group is set to the process group of the session leader.

The controlling terminal is inherited by a child process during a fork subroutine. A process cannot end the association with its controlling terminal by closing all of its file descriptors associated with the controlling terminal if other processes continue to have the terminal file open. A process that is not already the session leader or a group leader can break its association with its controlling terminal by using the setsid subroutine. Other processes remaining in the old session retain their association with the controlling terminal.

When the last file descriptor associated with a controlling terminal is closed (including file descriptors held by processes that are not in the controlling terminal’s session), the controlling terminal is disassociated from its current session. The disassociated controlling terminal can then be acquired by a new session leader.

A process can also remove the association it has with its controlling terminal by opening the tty file and issuing the following ioctl command:

ioctl (FileDescriptor, TIOCNOTTY, 0):

It is often useful to disassociate server processes from their controlling terminal so they cannot receive input from or be stopped by the terminal.

This device driver also supports the POSIX and Berkeley line disciplines.

Related Information
The fork subroutine, open subroutine, setsid subroutine.

The tty Subsystem Overview in AIX 5L Version 5.3 General Programming Concepts: Writing and Debugging Programs.
urandom and random Devices

Purpose
Source of secure random output.

Description
The /dev/random and /dev/urandom character devices provide cryptographically secure random output generated from interrupt timings or input written to the devices.

The /dev/random device is intended to provide high quality, cryptographically secure random output and will only return output for which sufficient (an equal or greater amount) random input is available to generate the output. If insufficient random input is available, reads from the /dev/random device will block until the request can be fulfilled unless the O_NONBLOCK flag was specified when the device was opened, in which case as much high quality output as could be generated is returned with the error code EAGAIN.

The /dev/urandom device provides a reliable source of random output, however the output will not be generated from an equal amount of random input if insufficient input is available. Reads from the /dev/urandom device always return the quantity of output requested without blocking. If insufficient random input is available, alternate input will be processed by the random number generator to provide cryptographically secure output, the strength of which will reflect the strength of the algorithms used by the random number generator. Output generated without random input is theoretically less secure than output generated from random input, so /dev/random should be used for applications for which a high level of confidence in the security of the output is required.

Data written to either device is added to the pool of stored random input and may be used for generating output. Writes behave identically for both devices and will not block.

Implementation Specifics
The /dev/random and /dev/urandom devices are created from major and minor numbers assigned by the device configuration subsystem when the random number generator is loaded, so the device names should always be used when attempting to locate or open the devices. The devices are deleted when the random number generator is unloaded. When the system is shut down using the shutdown command, output is taken from the /dev/urandom device and is written back to the /dev/random device when the random number generator is loaded on the next boot to provide starting entropy to the generator, enhancing the quality of the stored random input after boot.

Input is gathered from interrupt timings when the pool of stored random input falls below half full and continues to be gathered until the pool is again full. This process causes a minor performance impact to all external interrupts while timings are being gathered, which ceases when timings cease to be gathered. Data written to either of the random devices will also contribute to the pool of stored random input and can influence the output, thus writing to these devices should be a privileged operation. This is enforced by the permissions of the devices, so it can be changed by the administrator to be completely disallowed if desired.

vty_server Special File

Purpose
The virtual terminal server (vty_server or vts) is an AIX device driver used to create a tty-style connection from the partition on which the virtual terminal server is running to a virtual terminal (not a virtual terminal server) on another partition.
Description
The vty has a master side (vty server) and a slave side (vty). The vty server is opened and used by special programs such as mkvtterm. The vty is opened and used by shells such as ksh. The data path is bidirectional.

vts_ioctl
The vts_ioctl command is called via the device switch table when an application calls ioctl with a file descriptor that was returned from a successful open of a vts port. vts_ioctl performs specialized control operations on the port.

Syntax:
```c
struct vts_ioctl {
    void *vi_buffer; /* pointer to user's buffer */
    int vi_bufsize; /* size of user's buffer */
    int vi_offset; /* offset into data */
    int vi_result; /* bytes transferred */
};
```

The vi_buffer parameter is a pointer to a buffer in user space. The vi_bufsize parameter specifies the size of the buffer. For VTS_READ_CLCS and VTS_READ_PARTNER, data is moved from the vts driver into the buffer specified by vi_buffer. For VTS_WRITE_PARTNER, data is pulled from the buffer specified by vi_buffer and moved into the vts driver.

The vi_offset parameter is used only for VTS_READ_CLCS. An application can make multiple VTS_READ_CLCS ioctl calls; the driver may have more data to transfer to the application than will fit in the buffer that is specified by the vi_buffer parameter.

The vi_offset is set to 0 for the first FTS_READ_CLCS. This causes the driver to query PHYP for the current list of possible CLCs. The first sequence of the CLCs is moved into user space. Each CLC is separated by a new line and the last CLC is terminated by a null. A count of the number of bytes moved into user space (excluding the terminating null) is stored in vi_result.

The application determines how the data is processed (for example, the application may save the data in a linked list). vi_result is then added to vi_offset and another VTS_READ_CLCS is performed. This causes the driver to move the next set of CLCs into the application’s buffer. This sequence of operations continues until zero is returned in vi_result.

A sequence of VTS_READ_CLCS calls results in a sequence of CLCs separated by new lines and terminated by a null. The following are the possible error conditions:

EIO A driver call to PHYP returned with an error.
ENOMEM No memory is available to hold the CLC list.
EAGAIN The copyin or copyout failed.

A VTS_READ_PARTNER call returns the value that was passed on the last successful VTS_WRITE_PARTNER call. If the vts is not currently connected, a null string is returned. The only possible errors are errors that are returned by copyin or copyout routines.

The VTS_WRITE_PARTNER call creates and ends connections. The following are possible errors when ending a connection:

ENXIO The vts is not currently connected.
The `H_FREE_VTERM` call to PHYP returned `H_HARDWARE`. This usually indicates that there is a problem with PHYP or that the partition connection failed.

The `H_FREE_VTERM` call to PHYP returned `H_PARAMETER`. This usually indicates a driver problem.

To create a connection, the buffer must contain a CLC that was passed from `VTS_READ_CLCs`. The CLC can contain the ending new line, which is removed. The following are possible errors:

**EBUSY**

The vts is already connected. Use `VTS_READ_PARTNER` to determine where the vts is connected, or use `VTS_WRITE_PARTNER` with a null name to end the connection.

**EINVAL**

The CLC that was passed is not in the list of valid connections for this vts. EINVAL is also returned if the `H_REGISTER_VTERM` call to PHYP returns `H_PARAMETER`.

**Note:** If a reconfiguration has occurred, the list of valid CLCs may have changed.

The `H_REGISTER_VTERM` call to PHYP returned `H_PARAMETER`. This may be due to a failed PHYP.

**Related Information**

The `ioctl` subroutine, `ttyname` subroutine.

TTY Subsystem in *AIX 5L Version 5.3 General Programming Concepts: Writing and Debugging Programs*.

TTY Drivers in *AIX 5L Version 5.3 General Programming Concepts: Writing and Debugging Programs*.

---

**x25sn Special File**

**Purpose**

Provides access to the X.25 Interface Co-Processor/2 adapters by way of the X.25 Interface Co-Processor/2 device handler.

**Description**

The `x25sn` character special file provides access to the X.25 Interface Co-Processor/2 device handler, which provides access to a X.25 packet switching network. The device handler supports up to four X.25 Interface Co-Processor/2 adapters.

**Usage Considerations**

When accessing the X.25 Interface Co-Processor/2 device handler, the following should be taken into account:

**Driver initialization and termination**

The device handler may be loaded and unloaded. The device handler supports the configuration calls to initialize and terminate itself.

**Special file support**

Calls other than the `open` and `close` subroutines are discussed based on the mode in which the device handler is operating.
Subroutine Support
The X.25 Interface Co-Processor/2 device handler provides specific support for the open, close, read, write, and ioctl subroutines.

**open and close Subroutines:** The device handler supports the /dev/x25sn special file as a character-multiplex special file. The special file must be opened for both reading and writing (O_RDWR). There are no particular considerations for closing the special file. The special file name used in an open call differs depending upon how the device is to be opened. For each of the following types of special files, the value \( n \) is \( 0 \leq n \leq 7 \):

- /dev/x25sn: Starts the device handler on the next available port.
- /dev/x25sn/D: Opens the device handler for the specified port in Diagnostic mode.
- /dev/x25sn/M: Opens the device handler for reading and writing data to the monitor facilities on the X.25 Interface Co-Processor/2.
- /dev/x25sn/R: Opens the device handler for updating the routing table.

**read Subroutine:** Can take the form of a read, readx, readv, or readvx subroutine. For this call, the device handler copies the data into the buffer specified by the caller.

**write Subroutine:** Can take the form of a write, writex, writev, or writevx subroutine. For this call, the device handler copies the user data into a kernel buffer and transmits the data on the network.

**ioctl Subroutine:** The device handler supports the following ioctl operations:

- CIO_DNLD: Downloads a task.
- CIO_GET_STAT: Gets device statistics.
- CIO_HALT: Halts a session.
- CIO_QUERY: Queries a device.
- CIO_START: Starts a session.
- IOCINFO: Identifies a device.
- X25_ADD_ROUTER_ID: Adds a router ID.
- X25_COUNTER_GET: Gets a counter.
- X25_COUNTER_READ: Reads the contents of a counter.
- X25_COUNTER_REMOVE: Removes a counter from the system.
- X25_COUNTER_WAIT: Waits for the contents of counters to change.
- X25_DELETE_ROUTER_ID: Deletes a router ID.
- X25_DIAG_MEM_READ: Reads memory from the X.25 Interface Co-Processor/2 into a user’s buffer.
- X25_DIAG_MEM_WRITE: Writes memory to the X.25 Interface Co-Processor/2 from a user’s buffer.
- X25_DIAG_TASK: Provides the means to download the diagnostics task on to the card.
- X25_LINK_CONNECT: Connects a link.
- X25_LINK_DISCONNECT: Disconnects a link.
- X25_LINK_STATUS: Returns the status of the link.
- X25_LOCAL_BUSY: Enables or disables receiving of data packets on a port.
- X25_REJECT: Rejects a call.
- X25_QUERY_ROUTER_ID: Queries a router ID.
- X25_QUERY_SESSION: Queries a session.
Error Conditions
The following error conditions may be returned when accessing the device handler through the `/dev/x25s{n}` special file:

- **EACCES**: Indicates that the call application does not have the required authority.
- **EAGAIN**: Indicates there were no packets to be read or the transmit queue is full, and the device was opened with the DNDELAY flag set.
- **EBUSY**: Indicates that the device was busy and could not accept the operation.
- **EFAULT**: Indicates that an invalid address was specified.
- **EIDRM**: Indicates that the counter has been removed.
- **EINVAL**: Indicates that the device was opened with the DNDELAY flag set.
- **EIO**: Indicates that an error has occurred. The status field in the status-control block contains more information.
- **EMSGSIZE**: Indicates that the data to be given to the user was greater than the length of the buffer specified. The data in the buffer is truncated.
- **ENOBUFS**: Indicates that no buffers are available.
- **ENODEV**: Indicates that the device requested does not exist.
- **ENOMEM**: Indicates that the X.25 device handler was unable to allocate space required for the open.
- **ENOSPC**: Indicates that there are no counters available to allocate.
- **ENXIO**: Indicates that the device was not completely configured. Initial configuration must be completed before any starts can be issued.
- **EPERM**: Indicates the user does not have permission to perform the requested operation.

Related Information
The `close` subroutine, `open` subroutine, `read` or `readx` subroutine, `write` or `writex` subroutine.
Chapter 4. Header Files

Information that is needed by several different files or functions is collected into a header file. A header file contains C-language definitions and structures. Centralizing information into a header file facilitates the creation and update of programs. Because `#include` statements are used to insert header files into a C-language program, header files are often referred to as include files.

Header files define the following functions:
- Structures of certain files and subroutines
- Type definition (typedef) synonyms for data types
- System parameters or implementation characteristics
- Constants and macros that are substituted during the C language preprocessing phase.

By convention, the names of header files end with `.h` (dot h). The `.h` suffix is used by header files that are provided with the operating system; however, the suffix is not required for user-generated header files.

**Note:** Several of the header files provided with the operating system end with `.inc` (include file).

Additional header files are provided with the operating system. Most of these can be found in either the `/usr/include` directory or the `/usr/include/sys` directory. Use the `pg` command to view the contents of a header file.

More information about the following header files is provided in this documentation:
- `a.out.h` Defines the structure of the standard `a.out` file.
- `acct.h` Describes the format of the records in the system accounting files.
- `ar.h` Describes the format of an archive file.
- `audit.h` Defines values used by the auditing system as well as the structure of a bin.
- `core.h` Describes the structures created as a result of a core dump.
- `ct_ffdc.h` Provides data types, definitions, and interface prototypes for the First Failure Data Capture (FFDC) C language library interfaces.
- `dirent.h` Describes the format of a file system-independent directory entry.
- `eucioctl.h` Defines ioctl operations and data types for handling EUC code sets.
- `fcntl.h` Defines values for the `fcntl` and `open` subroutines.
- `filsys.h` Contains the format of a file system logical volume.
- `flock.h` Defines the file control options.
- `fullstat.h` Describes the data structure returned by the `fullstat` and `ffullstat` subroutines.
- `iconv.h` Defines types, macros, and subroutines for character code set conversion.
- `ipc.h` Defines structures used by the subroutines that perform interprocess communications operations.
- `ldr.h` Describes the `ld_info` data type and loader entry points.
- `libperfstat.h` Describes the structures and constants used by the `libperfstat` API subroutines.
- `limits.h` Defines implementation limits identified by the IEEE POSIX 1003 standard.
- `math.h` Defines math subroutines and constants
- `mode.h` Defines the interpretation of a file mode.
- `msg.h` Defines structures used by the subroutines that perform message queueing operations.
- `mtio.h` Defines the magnetic tape user include file.
- `param.h` Defines certain hardware-dependent parameters.
- `poll.h` Defines the `pollfd` structure used by the `poll` subroutine.
- `sem.h` Defines the structures that are used by subroutines that perform semaphore operations.
- `sgtty.h` Defines structures used by the Berkeley terminal interface.
- `shm.h` Defines structures used by the subroutines that perform shared memory operations.
- `spc.h` Defines external interfaces provided by the System Resource Controller (SRC) subroutines.
- `srcobj.h` Defines structures used by the System Resource Controller (SRC) subsystem.
Describes the data structure returned by the status subroutines. Describes the structure of the statistics returned by the status subroutines. Describes the structure of the statistics that are returned by the statvfs subroutines and fsatvfs subroutines. Defines the _system_configuration structure. Defines structures used by the terminal interface for compatibility of Version 2 of the operating system. Defines structures used by the POSIX terminal interface. Defines the structure of the termios file, which provides the extended terminal interface. Defines primitive system data types. Defines POSIX implementation characteristics. Defines the format of certain user and accounting information files. Defines hardware-dependent values. Describes the structure of a mounted file system. Contains the structures used by the X.25 application programming interface.

Related Information

The `pg` command.

[Fatal Formats Overview](#) defines and describes file formats in general and lists file formats discussed in this documentation.

[Fatal Special Files Overview](#) defines and describes special files in general and lists special files discussed in this documentation.

List of Major Control Block Header Files

The Base Operating System constants and control block structure definitions are contained in header files in the `/usr/include` and `/usr/include/sys` directories. The following are the major constants and control blocks and their corresponding header files:

- `/usr/include/a.out.h` Common Object File Format (COFF) structures
- `/usr/include/core.h` An include file for the `/usr/include/sys/core.h` header file
- `/usr/include/errno.h` An include file for the `/usr/include/sys/errno.h` header file
- `/usr/include/lvmrec.h` LVM record structure
- `/usr/include/sgty.h` Line discipline structures and constants for Berkeley compatibility
- `/usr/include/signal.h` An include file for the `/usr/include/sys/signal.h` header file
- `/usr/include/termio.h` An include file for the `/usr/include/sys/termio.h` header file
- `/usr/include/termios.h` POSIX line-discipline structures and constants
- `/usr/include/xcoff.h` Extended Common Object File Format structures
- `/usr/include/sys/acct.h` Accounting structures
- `/usr/include/sys/bdisk.h` Bus-attached-disk structures
- `/usr/include/sys/bbdir.h` Bad-block directory structure
- `/usr/include/sys/bootrecord.h` Boot record structure
- `/usr/include/sys/buf.h` Buffer header structures
- `/usr/include/sys/cdrom.h` CD-ROM structures
- `/usr/include/sys/cfgodm.h` Configuration object class structures
- `/usr/include/sys/configrec.h` Disk configuration record structure
- `/usr/include/sys/core.h` Core dump structure
- `/usr/include/sys/debug.h` Traceback table or procedure-end table
- `/usr/include/sys/device.h` Device switch table
- `/usr/include/sys/deviceq.h` Device queue-management structures
- `/usr/include/sys/devinfo.h` Device information structures
/usr/include/sys/dir.h Directory entry structures
/usr/include/sys/display.h Virtual display driver structures
/usr/include/sys/dump.h Component dump table structure
/usr/include/sys/entuser.h Ethernet device driver structures
/usr/include/sys/errids.h Error-log record identifiers
/usr/include/sys/erno.h Error codes
/usr/include/sys/fd.h Diskette device driver structures
/usr/include/sys/file.h File structure
/usr/include/sys/fstypes.h File-system parameter table
/usr/include/sys/hd_psn.h Layout of reserved space on the disk
/usr/include/sys/idc.h IDE device driver structures
/usr/include/sys/inode.h I-node structures
/usr/include/sys/intr.h Interrupt handler structures
/usr/include/sys/ipc.h Interprocess Communications (IPC) structures
/usr/include/sys/iplcb.h Initial Program Load (IPL) control block structure
/usr/include/sys/idr.h Loader structures and constants
/usr/include/sys/low.h Kernel Page 0 definition
/usr/include/sys/machine.h Machine dependent control registers
/usr/include/sys/mbuf.h Memory buffer structures
/usr/include/sys/mdio.h Machine device driver structures
/usr/include/sys/mount.h Mount structures
/usr/include/sys/mpqpp.h Multiprotocol Quad Port (MPQP) device-driver structures
/usr/include/sys/msg.h Message queue structures
/usr/include/sys/mstsave.h Machine State Save Area structures
/usr/include/sys/param.h Process management constants
/usr/include/sys/pri.h Constants for process priorities
/usr/include/sys/proc.h Process table structure
/usr/include/sys/pseg.h Process private segment layout
/usr/include/sys/reg.h Machine-dependent registers
/usr/include/sys/scdisk.h SCSI-disk device driver structures
/usr/include/sys/scsi.h SCSI device driver structures
/usr/include/sys/seg.h Memory management constants
/usr/include/sys/sem.h Semaphore structures
/usr/include/sys/shm.h Shared-memory facility structures
/usr/include/sys/signal.h Signal structures and constants
/usr/include/sys/socketvar.h Sockets structures
/usr/include/sys/stat.h File status structure
/usr/include/sys/system.h System global declarations
/usr/include/sys/termio.h Line discipline structures and constants for compatibility of Version 2 of the operating system
/usr/include/sys/timer.h Timer structures
/usr/include/sys/tokuser.h Token-ring device handler structures
/usr/include/sys/trchkid.h Trace hook IDs
/usr/include/sys/user.h User structure or user area (ublock)
/usr/include/sys/utsname.h UTSNAME structure (system name, node ID, machine ID)
/usr/include/sys/var.h Runtime system parameter structure
/usr/include/sys/vfs.h Virtual file system structures
/usr/include/sys/vnode.h Virtual i-node (v-node) structures
/usr/include/sys/xcoff.h Extended Common Object File Format structures
/usr/include/sys/xmalloc.h Heap structure
/usr/include/sys/xmem.h Cross memory service structures
ct_ffdc.h File

Purpose
Provides data types, definitions, and interface prototypes for the First Failure Data Capture (FFDC) C language library interfaces.

Description
This header file must be included by any C and C++ language source code files that make use of the First Failure Data Capture C language interfaces. This file contains the C language prototypes for the First Failure Data Capture interfaces, the symbolic constants used as return codes from these interfaces, and data type definitions needed by First Failure Data Capture C and C++ language clients.

C Language Interface Selection Control
This file provides the compiler definition FC_VERSION. This definition controls which version of the First Failure Data Capture interfaces should be used during compilation of the source code. Currently, only one version of the First Failure Data Capture interfaces are available, and the value of FC_VERSION is set to a default value of 1. Future versions of the First Failure Data Capture interfaces can be used during compilation—when they become available—by setting the value for FC_VERSION on the compilation command line. If this variable is not set during compilation, the value for FC_VERSION reverts to the default value of 1, and the initial version of the FFDC interfaces is used.

Data Types
The fc_eid_t data type defined by this module is used to store a First Failure Data Capture Failure Identifier. This identifier is created by the fc_push_stack and fc_log_error interfaces whenever these interfaces are successful in recording failure information. This identifier contains information in an encoded form to indicate the system on which the record was made, the time when the record was made, and the location of the record. First Failure Data Capture commands such as fcreport and fcstkprt can be used at a later time to obtain the exact failure report for problem determination efforts.

FFDC Environment Establishment Codes
A First Failure Data Capture client application uses the fc_init interface to specify how the FFDC Environment should be established. The following selections are supported:

FC_LOG
A basic FFDC Environment is established, which permits the application to record failure information to the AIX Error Log and the BSD System Log. An FFDC Error Stack is not established for use by the application in this case, unless this value is combined with either the FC_STACK_CREAT or the FC_STACK_INHERIT options described below. This selection would be used by applications making use of the fc_log_error interface only.

FC_STACK_CREAT
Creates an FFDC Error Stack Environment if one was not previously created by an ancestor of this process, or inherits the FFDC Error Stack Environment if an ancestor previously established one. The FFDC Error Stack Environment permits the application to record information to the AIX Error Log, the BSD System Log, and the FFDC Error Stack. This selection is used by applications that wish to use the fc_push_stack interface as well as the fc_log_error interface. This selection is not to be combined with the FC_STACK_INHERIT option described next.

FC_STACK_INHERIT
Inherit an FFDC Error Stack Environment only if an ancestor process previously established an FFDC Error Stack Environment. If an ancestor did not establish such an environment, the application does not make use of an FFDC Error Stack, but may still make use of the AIX Error Log and the BSD System Log. Do not combine this selection with the FC_STACK_CREAT option specified previously.
Record Type Definitions
Seven FFDC Event Types are specified in this file. These event types are used to instruct the fc_log_error interface as to the severity of the condition being logged:

**FFDC_EMERG**
A severe failure has occurred, and the system is in danger of coming offline. This information is required by the system administrator to bring the system back online. The AIX Error Log type equivalent is PEND. The BSD System Log priority equivalent is **LOG_EMERG**.

**FFDC_ERROR**
A permanent failure has occurred, and the condition will persist until it is repaired. The system is not in danger of coming offline. The AIX Error Log type equivalent is PERM. The BSD System Log priority equivalent is **LOG_ERR**.

**FFDC_STATE**
An event of some significance has occurred, but the event does not constitute a failure condition. The AIX Error Log type equivalent is INFO. The BSD System Log priority equivalent is **LOG_NOTICE**.

**FFDC_PERF**
A condition has been noticed which can or will degrade the system’s performance below acceptable levels. The system is not in danger of coming offline, but performance may be unacceptably slow, which can result in random failures in system applications, such as timeout conditions. The AIX Error Log type equivalent is PERF. The BSD System Log priority equivalent is **LOG_WARNING**.

**FFDC_TRACE**
This entry identifies the name and location of a trace file generated by an application or system component. Such an entry would be made when a trace has been enabled in an application or a component, to indicate where the trace file resides. The AIX Error Log type equivalent is UNKN. The BSD System Log priority equivalent is **LOG_INFO**.

**FFDC_RECOV**
A recovery action has been successfully completed by the system in response to an FFDC_EMERG, FFDC_ERROR, or FFDC_PERF condition. Such an entry would be created only after an FFDC_EMERG, FFDC_ERROR, or FFDC_PERF condition was detected, and a recovery action started in response to that condition completed successfully. The AIX Error Log type equivalent is TEMP. The BSD System Log priority equivalent is **LOG_DEBUG**.

**FFDC_DEBUG**
A failure condition was detected. Unlike the FFDC_ERROR case, the failure is not a permanent condition, or the system can continue successfully with the condition present. The AIX Error Log type equivalent is UNKN. The BSD System Log priority equivalent is **LOG_DEBUG**.

Examples
To use this file in a C or C++ language program, add the following line near the beginning of the source code module:

```c
#include <rsct/ct_ffcd.h>
```

Location
/usr/sbin/rsct/include/ct_ffdc.h
/usr/include/rsct/ct_ffdc.h
Related Information
The fcdispfid, fcdecode, fcreport and fcstkrpt commands.

The fc_display_fid, fc_eid_init, fc_eid_is_set, fc_init, fc_log_error, and fc_push_stack subroutines.

The RSCT First Failure Data Capture Programming Guide and Reference.

dirent.h File

Purpose
Describes the format of a file system-independent directory entry.

Description
The /usr/include/dirent.h file describes the format of a directory entry without reference to the type of underlying system.

The dirent structure, defined in the dirent.h file, is used for directory access operations. Using these access operations and the dirent structure, along with its associated constants and macros, shields you from the details of implementing a directory and provides a consistent interface to directories across all types of file systems.

The dirent structure contains the following fields for each directory entry:

ulong_t d_offset; /* actual offset of this entry */
ino_t d_ino; /* inode number of entry */
ushort_t d_reclen; /* length of this entry */
ushort_t d_namlen; /* length of string in d_name */
char d_name[_D_NAME_MAX+1]; /* name of entry (filename) */

_D_NAME_MAX is a constant that indicates the maximum number of bytes in a file name for all file systems. (Related to this constant is the PATH_MAX constant, which specifies the maximum number of bytes in the full path name of a file, not including the terminating null byte.)

The value of the _D_NAME_MAX constant is specific to each type of filesystem type. It can be determined by using the pathconf or fpathconf subroutine.

The size of a dirent structure depends on the number of bytes in the file name.

The _DNAME_MAX and PATH_MAX constants specify maximum file names and path names, respectively, across all types of file systems. The constants defined by a particular file system are applicable only to that file system. Using file system-specific constants and directory structures makes it very difficult to port code across different types of file systems.

Related Information
The dir file, sys/types.h file.

The pathconf or fpathconf subroutine.

Understanding JFS i-nodes in AIX 5L Version 5.3 General Programming Concepts: Writing and Debugging Programs explains how the operating system uses i-nodes.
dlfcn.h File

Purpose
Describes dynamic linking.

Syntax
#include <dlfcn.h>

Description
The <dlfcn.h> header defines at least the following macros for use in the construction of a dlopen mode argument:

- RTLD_LAZY: Relocations are performed at an implementation-dependent time.
- RTLD_NOW: Relocations are performed when the object is loaded.
- RTLD_GLOBAL: All symbols are available for relocation processing of other modules.
- RTLD_LOCAL: All symbols are not made available for relocation processing by other modules.

The header <dlfcn.h> declares the following functions, which may also be defined as macros:

- dlopen(const char *, int);
- dlsym(void *, const char *);
- dlclose(void *);
- dlerror(void);

Related Information
The dlopen, dlclose, dlsym, dlerror subroutines.

eucioctl.h File

Purpose
Defines ioctl operations and data types for handling EUC code sets.

Description
The eucioctl.h file contains information used for handling Extended UNIX Code (EUC) multibyte code sets. It consists of ioctl operations and the related data structure.

The eucioc structure contains the following fields:

- eucw[4]: Specifies the memory width of the code set. It indicates the number of bytes used to store the multibyte characters of each of the four classes.
- scrw[4]: Specifies the screen width of the code set. It indicates the number of columns used to display the multibyte characters of each of the four classes.

This structure is used in the following ioctl operations:

- EUC_WGET: Returns the EUC character widths. The eucioc structure is filled with the memory and screen widths of the current EUC code set.
- EUC_WSET: Sets the EUC character widths. The eucioc structure is used to set the memory and screen widths of the current EUC code set.

Related Information
The ioctl subroutine.
.readyState = 1;

try {
  // send request
  $.ajax({
    url: 'http://example.com/api',
    type: 'GET',
    data: {param1: 'value1', param2: 'value2'},
    success: function(response) {
      // handle response
      console.log(response);
    },
    error: function(xhr, status, error) {
      // handle error
      console.error(error);
    }
  });
} catch (error) {
  // handle exception
  console.error(error);
}
File Status Flags for open Subroutine
The following file status flags are accessible to the open subroutine:

- **O_DEFER** Deferred update
- **O_DELAY** Open with delay
- **O_DIRECT** Open for Direct I/O

File Descriptor Flags for fcntl Subroutine
The following file descriptor flag is accessible to the fcntl subroutine:

- **FD_CLOEXEC** Close this file during an exec.

File flag values corresponding to file access modes are as follows:

- **FREAD** File is open for read.
- **FWRITE** File is open for write.

Notes:
1. The **FREAD** and **FWRITE** flags cannot be used unless the _KERNEL flag has been defined.
2. The ldfcn.h file also assigns values to the **FREAD** and **FWRITE** options. If you use the ldfcn.h and fcntl.h files together, directly or indirectly, you should use the #undef statement on the **FREAD** and **FWRITE** options of one of the header files. If you do not, the compiler will return a warning about using duplicate definitions.

Command Values for fcntl Subroutine
The Command values for the fcntl subroutine (that is, for fcntl subroutine requests) are:

- **F_DUPFD** Duplicate the file description.
- **F_GETFD** Get the file description flags.
- **F_SETFD** Set the file description flags.
- **F_GETFL** Get the file status flags and file access modes.
- **F_SETFL** Set the file flags.
- **F_GETLK** Return information about an existing file lock.
- **F_GETLK64** Return information about an existing file lock.
- **F_SETLK** Set or clear a file lock.
- **F_SETLK64** Set or clear a file lock.
- **F_SETLKW** Set or clear a file lock and wait if blocked.
- **F_SETLKW64** Set or clear a file lock and wait if blocked.
- **F_GETOWN** Get the descriptor owner.
- **F_SETOWN** Set the descriptor owner.

Related Information
The fcntl subroutine, open, openx, or creat subroutine.

The sys/types.h file, unistd.h file.

The Header Files Overview defines header files, describes how they are used, and lists several header files for which information is provided.
filsys.h File

Purpose
Contains the format of a Journaled File System (JFS) logical volume.

Syntax
#include <sys/filsys.h>

Description
The filsys.h file contains the format of a JFS file system. A JFS file system has a common format for vital information and is divided into a number of fixed-sized units, or fragments. Fragments serve as the basic unit of file system disk space allocation and can be smaller than the file system logical block size, which is 4096 bytes. The file system superblock records the logical block size and fragment size, as well as the size of the entire file system.

A unique feature of the JFS is the implementation of file system metadata as unnamed files that reside in that file system. For example, the disk i-nodes for any file system are contained in the blocks fragments allocated to the file described by the INODES_I i-node. The i-node number for the boot file is 0. Each of the following reserved i-nodes corresponds to a file system metadata file:

  SUPER_I    Superblock file
  INODES_I   Disk i-nodes
  INDIR_I    Indirect file blocks, double and single
  INOMAP_I   i-node allocation bit map
  ROOTDIR_I  Root directory i-node
  DISKMAP_I  Block Fragment allocation bit map
  INODEX_I   i-node extensions
  INODEXMAP_I Allocation map for i-node extensions

The first 4096 bytes of the file system are unused and available to contain a bootstrap program or other information. The second 4096 bytes of the file system are used to hold the file system superblock. The structure of a JFS superblock follows:

/* The following disk-blocks are formatted or reserved:
*  *
*    ipl block 0 - not changed by filesystem.
*  *
*    superblocks at 1 x 4096 (primary superblock) and 31 x
*    4096 (secondary superblock). the secondary super-block
*    location is likely to be on a different disk-surface than
*    the primary super-block. both structures are allocated as
*    fragments in ".superblock".
*  *
*    fragments for .inodes according to BSD layout. each
*    allocation group contains a fixed number of disk inodes.
*    for fsv3 file systems, each allocation group contains one
*    inode per 4096 byte fragment of the allocation group,
*    with the number of fragments within each group described
*    by the s_agysize field of the superblock. for fsv3p file
*    systems, the number of inodes per group is described by
*    the s_iagsize field of the superblock and may be less
*    than or greater than the number of fragments per group.
*    for these file systems, s_agysize describes the number of
*    s_fragsize fragments contained within each allocation
*    group.
*  *
*    the first allocation group inodes starts at 32 x
*    4096 bytes and consumes consecutive fragments sufficient
to hold the group's inodes. The inode fragments for all
other allocation groups start in the first fragments of
the allocation group and continue in consecutive
fragments sufficient to hold the group's inodes.

other fragments are allocated for .indirect, .diskmap,
.inodemap, and their indirect blocks starting in the
first allocation-group.

The special fs inodes formatted and their usage is as follows:

inode 0 - never allocated - reserved by setting
n_link = 1
inode 1 - inode for .superblock
inode 2 - inode for root directory
inode 3 - inode for .inodes
inode 4 - inode for .indirect
inode 5 - inode for .inodemap - allocation map for
.inodes
inode 6 - inode for .diskmap - disk allocation map
inode 7 - inode for .inodex - inode extensions
inode 8 - inode for .inodexmap - allocation map for
.inodex
inode 9 - 16 - reserved for future extensions

except for the root directory, the special inodes are not in
any directory.

*/
#define IPL_B 0
#define SUPER_B 1
#define SUPER_B1 31
#define INODES_B 32
#define NON_B 0
#define SUPER_I 1
#define ROOTDIR_I 2
#define INODES_I 3
#define INDIR_I 4
#define INODES_I 5
#define DISKMAP_I 6
#define INODEX_I 7
#define INODESMAP_I 8

/*
* super block format. primary superblock is located in the
* second 4096 bytes of the file system.
* the secondary super-block is not used except for disaster
* recovery.
*/

struct superblock
{
    char s_magic[4];    /* magic number */
    char s_flag[4];     /* flag word (see below) */
    int s_agsize;      /* fragments per allocation group */
    int s_logserial;   /* serial number of log when fs mounted */
    daddr_t s_fsize;    /* size (in 512 bytes) of entire fs */
    short s_bsize;     /* block size (in bytes) for this
                        system */
    short s_spare;     /* unused. */
    char s_fname[6];   /* name of this file system */
    char s_fpack[6];   /* name of this volume */
    dev_t s_logdev;    /* device address of log */

    /* current file system state information, values change over
time */
}
The fields of the superblock structure have the following functions:

- **s_fname**: Specifies the name of the file system.
- **s_fpack**: Specifies the name of the volume on which the file system resides.
- **s_fsize**: Specifies the entire file system size in 512-byte units.
- **s_bsize**: Specifies the file-system logical block size in bytes.
- **s_ffragsize**: Specifies the file system fragment size and is only valid for fsv3p file systems. For fsv3 file systems, the file-system fragment size is logically defined as the file-system logical block size.
- **s_agsize**: Specifies the number of fragments per file system allocation group. For fsv3 file systems, this field also specifies the number of disk i-nodes per file system allocation group.
- **s_iagsize**: Specifies the number of disk i-nodes per file system allocation group for fsv3p file systems. The **s_iagsize** field is only valid for fsv3p file systems.
- **s_magic**: Specifies the file-system magic number and is used to validate file systems. The *magic number* is encoded as a 4-byte character string to make it possible to validate the superblock without knowing the byte order of the remaining fields. To check for a valid fsv3 superblock, use a condition similar to:
  
  ```c
  if (strncmp(sp->s_magic,fsv3magic,4) == 0)
  ```
  
  For fsv3p file systems, superblock validation is made by checking both the **s_magic** and **s_version** fields.
- **s_version**: Specifies the file-system version number and is only valid for fsv3p file systems. To check for a valid fsv3p superblock, use a condition similar to:
  
  ```c
  if (strncmp(sp->s_magic,fsv3pmagic,4) == 0 &&
       sp->s_version == fsv3pvers)
  ```
  
- **s_logdev**: Specifies the device ID of the file system log device.
- **s_logserial**: Records the serial number of the log device at the time the file system was last mounted as modifiable.
**s_fmod**
Contains a flag to indicate the cleanliness of the file system. Whenever a file system is mounted, this flag is checked and a warning message is printed if the s_fmod field is equal to nonzero. A file system whose s_fmod field is equal to 0 is very likely to be clean, and a file system whose s_fmod field is equal to 2 is likely to have problems. The s_fmod field is intended to be a three-state flag with the third state being a sticky state. The three states are:

- 0 = File system is clean and unmounted.
- 1 = File system is clean and mounted.
- 2 = File system was mounted dirty.

If you only mount and unmount the file system, the flag toggles back and forth between states 0 and 1. If you mount the file system while the flag is in state 1, the flag goes to state 2 and stays there until you run the `fsck` command. The only way to clean up a corrupted file system (change the flag from state 2 back to state 0) is to run the `fsck` command.

**s_ronly**
Contains a flag indicating that the file system is mounted read-only. This flag is maintained in memory only; its value on disk is not valid.

**s_time**
Specifies the last time the superblock of the file system was changed (in seconds since 00:00 Jan. 1, 1970 (GMT)).

### Related Information

- The **param.h** file format.
- The `fsck` command, `fsdb` command, `mkfs` command.
- The **File systems** in *Operating system and device management* explains file system types, management, and structure.
- The **Mounting** in *Operating system and device management* explains mounting files and directories, mount points, and automatic mounts.
- The **Logical volume storage** in *Operating system and device management* explains the Logical Volume Manager, physical volumes, logical volumes, volume groups, organization, ensuring data integrity, and understanding the allocation characteristics.

### flock.h File

**Purpose**
Defines file control options.

**Description**
The `flock` structure in the `/usr/include/sys/flock.h` file, which describes a lock, contains the following fields:
l_type  Describes the type of lock. If the value of the *Command* parameter to the *fcntl* subroutine is *F_SETLK* or *F_SETLKW*, the `l_type` field indicates the type of lock to be created. Possible values are:

- **F_RDLCK**  
  A read lock is requested.

- **F_WRLCK**  
  A write lock is requested.

- **F_UNLCK**  
  Unlock. An existing lock is to be removed.

If the value of the *Command* parameter to the *fcntl* subroutine is *F_GETLK*, the `l_type` field describes an existing lock. Possible values are:

- **F_RDLCK**  
  A conflicting read lock exists.

- **F_WRLCK**  
  A conflicting write lock exists.

- **F_UNLCK**  
  No conflicting lock exists.

`l_whence`  Defines the starting offset. The value of this field indicates the point from which the relative offset, the `l_start` field, is measured. Possible values are:

- **SEEK_SET**  
  The relative offset is measured from the start of the file.

- **SEEK_CUR**  
  The relative offset is measured from the current position.

- **SEEK_END**  
  The relative offset is measured from the end of the file.

These values are defined in the `unistd.h` file.

`l_start`  Defines the relative offset in bytes, measured from the starting point in the `l_whence` field.

`l_len`  Specifies the number of consecutive bytes to be locked.

`l_sysid`  Contains the ID of the node that already has a lock placed on the area defined by the *fcntl* subroutine. This field is returned only when the value of the *Command* parameter is *F_GETLK*.

`l_pid`  Contains the ID of a process that already has a lock placed on the area defined by the *fcntl* subroutine. This field is returned only when the value of the *Command* parameter is *F_GETLK*.

`l_vfs`  Specifies the file system type of the node identified in the `l_sysid` field.

Although the *flock* structure is used by application programs to make file lock requests, the extended *flock* structure, *struct efdock*, is used internally by the kernel. The *eflock* structure is identical to the *flock* structure in that it has the same fields. The differences are that the `l_len` and `l_start` fields are 64 bit integers.

The *flock64* structure in the `/usr/include/sys/flock.h` file, which describes a lock, contains the following fields:
1_type  Describes the type of lock. If the value of the Command parameter to the fcntl subroutine is F_SETLK or F_SETLKW, the 1_type field indicates the type of lock to be created. Possible values are:

F_RDLCK  A read lock is requested.
F_WRLCK  A write lock is requested.
F_UNLCK  Unlock. An existing lock is to be removed.

If the value of the Command parameter to the fcntl subroutine is F_GETLK, the 1_type field describes an existing lock. Possible values are:

F_RDLCK  A conflicting read lock exists.
F_WRLCK  A conflicting write lock exists.
F_UNLCK  No conflicting lock exists.

1_whence Defines the starting offset. The value of this field indicates the point from which the relative offset, the 1_start field, is measured. Possible values are:

SEEK_SET  The relative offset is measured from the start of the file.
SEEK_CUR  The relative offset is measured from the current position.
SEEK_END  The relative offset is measured from the end of the file.

These values are defined in theunistd.h file.

1_start  Defines the relative offset in bytes, measured from the starting point in the 1_whence field. This field is of the type off64_t.

1_len  Specifies the number of consecutive bytes to be locked. This field is of the type off64_t.

1_sysid  Contains the ID of the node that already has a lock placed on the area defined by the fcntl subroutine. This field is returned only when the value of the Command parameter is F_GETLK.

1_pid  Contains the ID of a process that already has a lock placed on the area defined by the fcntl subroutine. This field is returned only when the value of the Command parameter is F_GETLK.

1_vfs  Specifies the file system type of the node identified in the 1_sysid field.

Related Information

The unistd.h file.

The fcntl subroutine, lockfx, lock, or flock subroutine, open, openx, or creat subroutine.

Header Files Overview defines header files, describes how they are used, and lists several of the header files for which information is provided in this documentation.

fullstat.h File

Purpose

Defines the data structure returned by the fullstat subroutine.
Description
The /usr/include/sys/fullstat.h file defines the data structure returned by the fullstat and ffullstat subroutines. This file also defines the Command parameters used by the fullstat and ffullstat subroutines. The fullstat structure contains the following fields:

Note: Time is measured in seconds since 00:00:00 GMT, January 1, 1970.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>st_dev</td>
<td>ID of device containing a directory entry for this file.</td>
</tr>
<tr>
<td>st_ino</td>
<td>File serial number.</td>
</tr>
<tr>
<td>st_mode</td>
<td>The mode of the file, as defined in the /usr/include/sys/mode.h file.</td>
</tr>
<tr>
<td>st_nlink</td>
<td>Number of links to file.</td>
</tr>
<tr>
<td>st_uid</td>
<td>User ID of the owner of the file.</td>
</tr>
<tr>
<td>st_gid</td>
<td>Group ID of the file owner group.</td>
</tr>
<tr>
<td>st_rdev</td>
<td>ID of this device. This field is defined only for character or block special files.</td>
</tr>
<tr>
<td>st_size</td>
<td>File size in bytes.</td>
</tr>
<tr>
<td>st_atime</td>
<td>Time of last access.</td>
</tr>
<tr>
<td>st_mtime</td>
<td>Time of last data modification.</td>
</tr>
<tr>
<td>st_ctime</td>
<td>Time of last file status change.</td>
</tr>
<tr>
<td>st_blksize</td>
<td>Optimal block size for the file system.</td>
</tr>
<tr>
<td>st_blocks</td>
<td>Number of blocks actually allocated to the file.</td>
</tr>
<tr>
<td>st_vfstype</td>
<td>File-system type as defined in the vmount.h file.</td>
</tr>
<tr>
<td>fst_type</td>
<td>Type of v-node.</td>
</tr>
<tr>
<td>fst_vfs</td>
<td>Virtual file system ID.</td>
</tr>
<tr>
<td>fst_flag</td>
<td>Indicates whether directory or file is a virtual mount point.</td>
</tr>
<tr>
<td>fst_i_gen</td>
<td>Generation number of the i-node.</td>
</tr>
<tr>
<td>fst_reserved[8]</td>
<td>Reserved.</td>
</tr>
</tbody>
</table>

The following fields are maintained for source-level compatibility with previous versions of the operating system:

- fst_uid_rev_tag
- fst_gid_rev_tag
- fst_nid

Related Information
The mode.h, stat.h, statfs.h, types.h, vmount.h file.

The statx, stat, lstat, fstat, fullstat, or ffullstat subroutine.

grp.h File

Purpose
Describes group structure.

Syntax
#include <grp.h>

Description
The grp.h header declares the structure group that includes the following members:

- char *gr_name the name of the group
- gid_t gr_gid numerical group ID
- char **gr_mem pointer to a null-terminated array of character pointers to member names
The `gid_t` type is defined as described in the `sys/types.h` header file.

The following are declared as functions and may also be defined as macros. Function prototypes must be provided for use with an ISO C compiler.

```c
struct group *getgrgid(gid_t);
struct group *getgrnam(const char *);
int getgrgid_r(gid_t, struct group *, char *, size_t, struct group **);
int getgrnam_r(const char *, struct group *, char *, size_t, struct group **);
struct group *getgrent(void);
void endgrent(void);
void setgrent(void);
```

**Related Information**

The `getgrent`, `endgrent`, `getgrnam`, `getgrgid` and `getgrgid_r` subroutines.

The `types.h` header file.

---

**iconv.h File**

**Purpose**

Defines types, macros, and subroutines for character code set conversion.

**Description**

The `/usr/include/iconv.h` file defines types, subroutines, and macros used in character code-set conversion by the iconv family of subroutines and commands. The `iconv.h` file defines the `iconv_t` data type.

**Related Information**

The `genxlt` command, `iconv` command.

The `iconv` subroutine, `iconv_close` subroutine, `iconv_open` subroutine.

[National Language Support Overview for Programming](#) in AIX 5L Version 5.3 General Programming Concepts: Writing and Debugging Programs.

[Converters Overview for Programming](#) in AIX 5L Version 5.3 General Programming Concepts: Writing and Debugging Programs.


---

**inode.h File**

**Purpose**

Describes a file system file or directory entry as it is listed on a disk.

**Syntax**

```c
#include <sys/types.h>
#include <sys/ino.h>
```
Description

The inode file for an ordinary file or directory in a file system has the following structure defined by the sys/ino.h file format:

```c
struct dinode
{
    /* generation number */
    ulong di_gen;
    /* mode_t returned by stat() */
    /* format, attributes and permission bits */
    mode_t di_mode;

    /* number of links to file (if != 0, inode is available) */
    ushort di_nlink;

    /* accounting ID */
    ushort di_acct;

    /* user id of owner */
    uid_t di_uid;

    /* group id of owner */
    gid_t di_gid;

    /* size of file */
    off_t di_size;

    /* number of blocks actually used by file */
    ulong di_nblocks;

    /* time last modified */
    struct timestruc_t di_mtime_ts;

    /* time last accessed */
    struct timestruc_t di_atime_ts;

    /* time last changed inode */
    struct timestruc_t di_ctime_ts;

    /* defines for old time_t names */
    #define di_mtime di_mtime_ts.tv_sec
    #define di_atime di_atime_ts.tv_sec
    #define di_ctime di_ctime_ts.tv_sec

    /* extended access control information */
    long di_acl; /* acl pointer */
    #define ACL_INCORE (1<<31)
    ulong di_sec; /* reserved */

    /* spares */
    ulong di_rsrvd[5];

    /***** file type-dependent information *****/
    /* size of private data in disk inode is D_PRIVATE. */
    /* location and size of fields depend on object type. */
    #define D_PRIVATE 48

    union di_info
    {
        /* all types must fit within d_private */
        char d_private[D_PRIVATE];
        /* jfs regular file or directory */
        struct regdir
        {
            /* privilege vector-only for non-directory */
            struct
```


```c
{ ulong_di_offset;
 ulong_di_flags;
 define;PCL_ENABLED(1<<31)
 define PCL_EXTENDED(1<<30)
 define PCL_FLAGS
 (PCL_ENABLED|PCL_EXTENDED)
 } di_privingo;
 priv_t_di_priv;
 /* ACL templates - only for directory */
 struct
 { ulong_di_aclf;
  ulong_di_acld;
  { _di_aclingo;
  } _di_sec;
 } _di_file;

/* offsets of regular file or directory private data. */
# define di_rdaddr _di_info._di_file._di_rdaddr
#define di_vindirect _di_info._di_file._di_vindirect
#define di_rindirect _di_info._di_file._di_rindirect
#define di_privoffset _di_privinfo._di_privoffset
#define di_privflags _di_privinfo._di_privflags
#define di_priv _di_info._di_file._di_sec._di_priv
#define di_aclf _di_info._di_file._di_sec._di_aclinfo._di_aclf
#define di_acld _di_info._di_file._di_sec._di_aclinfo._di_acld

/* special file (device) */
 struct
 { dev_t_di_rdev; /
  }_di_dev;

/* offsets of special file private data. */
# define di_rdev _di_infor._di_dev._di_rdev
# define di_bnlastr _di_info._di_dev._di_bnlastr
# define di_dgp _di_info._di_dev._di_dgp
# define di_pino _di_info._di_dev._di_pino

/*
 * symbolic link: link is stored inode if its
 * length is less than D_PRIVATE. Otherwise like
 * regular file.
 */
 union
 { char _s_private[D_PRIVATE];
  struct regdir_s_symfile;
 } _di_sym;

/* offset of symbolic link private data */
# define di_symlink _di_info._di_sym._s_private

/*
 * data for mounted filesystem. kept in inode = 0
 * and dev = devt of mounted filesystem in inode table.
 */
 struct mountnode
 { struct inode *iplog; /*itab of log*/
  struct inode *ipinode; /*itab of .inodes*/
  struct inode *ipind; /*itab of .indirect*/
  struct inode *ipinomap; /*itab of inode map*/
  struct inode *ipdmap; /*itab of disk map*/
  struct inode *_ipsuper; /*itab of super blk*/
}
```

Chapter 4. Header Files 837
struct inode *ipinodex; /*itab of .inodex*/
struct jfsmount *jfsmnt; /* ptr to mount data*/
ushort _fperpage; /* frag per block */
ushort _agsize; /* frags per ag */
ushort _lagsize; /* inodes per ag */
} mt_info;

/*
 * data for mounted filesystem. kept in inode = 0
 * and dev = devt of mounted filesystem in inode table.
 */
struct mountnode
{
    struct inode *iplog; /*itab of log*/
    struct inode *ipinode; /*itab of .inodes*/
    struct inode *ipind; /*itab of .indirect*/
    struct inode *ipinomap; /*itab of inode map*/
    struct inode *ipdmap; /*itab of disk map*/
    struct inode *ipsuper; /*itab of super blk*/
    struct inode *ipinodex; /*itab of .inodex*/
    struct jfsmount *jfsmnt; /* ptr to mount data*/
    ushort _fperpage; /* frag per block */
    ushort _agsize; /* frags per ag */
    ushort _lagsize; /* inodes per ag */
    ushort _compress /* > 0 if data comp */
} mt_info;

/* offsets of MOUNT data */
#define di_iplog _di_info._mt_info._iplog
#define di_ipinode _di_info._mt_info._ipinode
#define di_ipind _di_info._mt_info._ipind
#define di_ipinomap _di_info._mt_info._ipinomap
#define di_ipdmap _di_info._mt_info._ipdmap
#define di_ipsuper _di_info._mt_info._ipsuper
#define di_ipinodex _di_info._mt_info._ipinodex
#define di_jfsmnt _di_info._mt_info._jfsmnt
#define di_fperpage _di_info._mt_info._fperpage
#define di_agsize _di_info._mt_info._agsize
#define di_lagsize _di_info._mt_info._lagsize
#define di_compress _di_info._mt_info._compress

/*
 * log info. kept in inode = 0 and dev = devt of
 * log device filesystem in inode table.
 */
struct lognode
{
    int _logptr /* page number end of log */
    int _logsize /* log size in pages */
    int _logend /* eor in page _logptr */
    int _logsync /* addr in last syncpt record */
    int _nextsync /* bytes to next logsyncpt */
    struct gnode * _logdgp; /* pointer to device gnode */
} di_log;

/* offsets of MOUNT data */
#define di_iplog _di_info._mt_info._iplog
#define di_ipinode _di_info._mt_info._ipinode
#define di_ipind _di_info._mt_info._ipind
#define di_ipinomap _di_info._mt_info._ipinomap
#define di_ipdmap _di_info._mt_info._ipdmap
#define di_ipsuper _di_info._mt_info._ipsuper
#define di_ipinodex _di_info._mt_info._ipinodex
#define di_jfsmnt _di_info._mt_info._jfsmnt
#define di_fperpage _di_info._mt_info._fperpage
#define di_agsize _di_info._mt_info._agsize
#define di_lagsize _di_info._mt_info._lagsize
#define di_compress _di_info._mt_info._compress
/ * log info. kept in inode = 0 and dev = devt of * log device filesystem in inode table. */
struct lognode {
    int _logptr    /* page number end of log */
    int _logsize   /* log size in pages */
    int _logend    /* eor in page _logptr */
    int _logsync   /* addr in last syncpt record */
    int _nextsync  /* bytes to next logsyncpt */
    struct gnode  * _logdgp; /* pointer to device gnode */
}_di_log;

/* offsets of LOG data */
#define  di_logptr   _di_info._di_log._logptr
#define  di_logsize  _di_info._di_log._logsize
#define  di_logend   _di_info._di_log._logend
#define  di_logsync  _di_info._di_log._logsync
#define  di_nextsync _di_info._di_log._nextsync
#define  di_logdgp   _di_info._di_log._logdgp
}_di_info;

Related Information

Directories and File systems in Operating system and device management.

inttypes.h File

Purpose
Contains fixed size integral types.

Syntax
#include <inttypes.h>

Description
The inttypes.h header includes definitions of, at least, the following types:

int8_t 8-bit signed integral type.
int16_t 16-bit signed integral type.
int32_t 32-bit signed integral type.
int64_t 64-bit signed integral type.
uint8_t 8-bit unsigned integral type.
uint16_t 16-bit unsigned integral type.
uint32_t 32-bit unsigned integral type.
uint64_t 64-bit unsigned integral type.
intptr_t Signed integral type large enough to hold any pointer.
uintptr_t Unsigned integral type large enough to hold any pointer.
ipc.h File

Purpose
Describes the structures that are used by the subroutines that perform interprocess communications operations.

Syntax
#include <sys/ipc.h>

Description
The ipc.h file defines the following symbolic constants, types, and structures:

Symbolic Constants:

<table>
<thead>
<tr>
<th>Symbolic Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPC_CREAT</td>
<td>create entry if key doesn't exist</td>
</tr>
<tr>
<td>IPC_EXCL</td>
<td>fail if key exists</td>
</tr>
<tr>
<td>IPC_NOWAIT</td>
<td>error if request must wait</td>
</tr>
<tr>
<td>IPC_PRIVATE</td>
<td>private key</td>
</tr>
<tr>
<td>IPC_RMID</td>
<td>remove identifier</td>
</tr>
<tr>
<td>IPC_SET</td>
<td>set options</td>
</tr>
<tr>
<td>IPC_STAT</td>
<td>get options</td>
</tr>
<tr>
<td>IPC_ALLO</td>
<td>Centery currently allocated</td>
</tr>
<tr>
<td>IPC_R</td>
<td>read or receive permission</td>
</tr>
<tr>
<td>IPC_W</td>
<td>write or send permission</td>
</tr>
<tr>
<td>IPC_ERROR</td>
<td>truncates a message if too long</td>
</tr>
<tr>
<td>SHM_SIZE</td>
<td>change segment size (shared mem only)</td>
</tr>
</tbody>
</table>

The structure `ipc_perm` contains the following members:

<table>
<thead>
<tr>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>uid_t</td>
<td>owner's user id</td>
</tr>
<tr>
<td>gid_t</td>
<td>owner's group id</td>
</tr>
<tr>
<td>uid_t</td>
<td>creator's user id</td>
</tr>
<tr>
<td>gid_t</td>
<td>creator's group id</td>
</tr>
<tr>
<td>mode_t</td>
<td>access modes</td>
</tr>
<tr>
<td>unsigned short</td>
<td>seq</td>
</tr>
<tr>
<td>key_t</td>
<td>key</td>
</tr>
</tbody>
</table>

The types `uid_t, gid_t, mode_t, and key_t` are as defined in `<sys/types.h>`.

The following is declared as a function:

```c
key_t ftok(const char *, int);
```

Related Information
The `<types.h>` file.

The `ftok` subroutine.

iso646.h File

Purpose
Provides alternate spellings.

Syntax
#include <iso646.h>
Description

The iso646.h header file defines the following eleven macros (on the left) that expand to the corresponding tokens (on the right):

- `and` &
- `and_eq` &=
- `bitand` &
- `bitor` |
- `compl` ~
- `not` !
- `not_eq` !=
- `or` |
- `or_eq` |=
- `xor` ^
- `xor_eq` ^=

ldr.h File

Purpose

Describes the ld_info and ld_xinfo data types, along with associated loader entry points.

Syntax

```
#include <sys/ldr.h>
```

Description

The /usr/include/sys/ldr.h header file contains declarations of the ld_info and ld_xinfo data structures and the system loader entry points available to processes and kernel extensions.

The ld_info structure describes a loadable module in the context of either tracing a process (with the ptrace system call) or examining a core file. The ldr.h file can define 2 variants of the ld_info structure, one for describing 32-bit processes (__ld_info32) and one for describing 64-bit processes (__ld_info64). If the __LDINFO_PTRACE32__ symbol is defined, so is the struct __ld_info32 type. If the __LDINFO_PTRACE64__ symbol is defined, so is the struct __ld_info64 type. If the compilation mode is 32-bit and the __LDINFO_PTRACE32__ symbol is defined, the struct __ld_info32 and struct ld_info types are equivalent. If the compilation mode is 64-bit and the __LDINFO_PTRACE64__ symbol is defined, the struct __ld_info64 and struct ld_info types are equivalent.

When using ptrace in a 32-bit program to debug a 64-bit process, define __LDINFO_PTRACE64__. When using ptrace in a 64-bit program to debug a 32-bit process, define __LDINFO_PTRACE32__.

The types and sizes of these structures' fields depend on whether the compilation mode is 32-bit or 64-bit. The same field names are generated in both structure modes, with the exception that the 64-bit structure has an ldinfo_flags field.

The __ld_info32 and __ld_info64 structures contain the following fields of the indicated sizes and types; when two types are listed, the first is used when the compilation mode is 32-bit and the second is used when the mode is 64-bit:

<table>
<thead>
<tr>
<th>Field and Description</th>
<th>__ld_info32</th>
<th>__ld_info64</th>
</tr>
</thead>
<tbody>
<tr>
<td>ldinfo_next</td>
<td>Offset from current entry of next entry, or zero if last entry.</td>
<td>• Size: 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Type: uint</td>
</tr>
<tr>
<td>Field and Description</td>
<td>__ld_info32</td>
<td>__ld_info64</td>
</tr>
<tr>
<td>-----------------------</td>
<td>------------</td>
<td>------------</td>
</tr>
<tr>
<td>ldinfo_flags</td>
<td>Flags.</td>
<td>Flags: 4</td>
</tr>
<tr>
<td></td>
<td>Size: N/A</td>
<td>Size: 4</td>
</tr>
<tr>
<td></td>
<td>Type: N/A</td>
<td>Type: uint</td>
</tr>
<tr>
<td>ldinfo_fd</td>
<td>File descriptor returned by ptrace to debugger.</td>
<td>File descriptor returned by ptrace to debugger.</td>
</tr>
<tr>
<td></td>
<td>Size: 4</td>
<td>Size: 4</td>
</tr>
<tr>
<td></td>
<td>Type: int</td>
<td>Type: int</td>
</tr>
<tr>
<td>ldinfo_core</td>
<td>Offset into core file of object. (Overlaps the ldinfo_fd field.)</td>
<td>Offset into core file of object. (Overlaps the ldinfo_fd field.)</td>
</tr>
<tr>
<td></td>
<td>Size: 4</td>
<td>Size: 8</td>
</tr>
<tr>
<td></td>
<td>Type: int</td>
<td>Type: long long</td>
</tr>
<tr>
<td>ldinfo_textorg</td>
<td>Effective address of the loaded program image, including the XCOFF headers.</td>
<td>Effective address of the loaded program image, including the XCOFF headers.</td>
</tr>
<tr>
<td></td>
<td>Size: 4</td>
<td>Size: 8</td>
</tr>
<tr>
<td></td>
<td>Type: void *, uint</td>
<td>Type: unsigned long long, void *</td>
</tr>
<tr>
<td>ldinfo_textsize</td>
<td>Length of loaded program image.</td>
<td>Length of loaded program image.</td>
</tr>
<tr>
<td></td>
<td>Size: 4</td>
<td>Size: 8</td>
</tr>
<tr>
<td></td>
<td>Type: int</td>
<td>Type: long long</td>
</tr>
<tr>
<td>ldinfo_dataorg</td>
<td>Effective address of the start of data.</td>
<td>Effective address of the start of data.</td>
</tr>
<tr>
<td></td>
<td>Size: 4</td>
<td>Size: 8</td>
</tr>
<tr>
<td></td>
<td>Type: void *, uint</td>
<td>Type: unsigned long long, void *</td>
</tr>
<tr>
<td>ldinfo_datasize</td>
<td>Size of data, including the .bss section.</td>
<td>Size of data, including the .bss section.</td>
</tr>
<tr>
<td></td>
<td>Size: 4</td>
<td>Size: 8</td>
</tr>
<tr>
<td></td>
<td>Type: int</td>
<td>Type: long long</td>
</tr>
<tr>
<td>ldinfo_filename</td>
<td>Nul-terminated path name followed by nul-terminated member name; member name is empty if not an archive.</td>
<td>Nul-terminated path name followed by nul-terminated member name; member name is empty if not an archive.</td>
</tr>
<tr>
<td></td>
<td>Size: variable</td>
<td>Size: variable</td>
</tr>
</tbody>
</table>

The __ld_xinfo structure is similar to the __ld_info structure, but the __ld_xinfo structure has additional fields for modules defining thread-local storage. Furthermore, all fields are large enough to describe regions of a 64-bit process, so only a single variant of the __ld_xinfo structure exists.

The additional fields in the __ld_xinfo structure are the following:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ldinfo_tdata</td>
<td>Size of the .tdata section.</td>
</tr>
<tr>
<td>ldinfo_tbss</td>
<td>Size of the .tbss section.</td>
</tr>
<tr>
<td>ldinfo_data</td>
<td>Effective address of the initialization template for .tdata.</td>
</tr>
<tr>
<td>ldinfo_dataoff</td>
<td>Offset of the module's thread-local storage in its region.</td>
</tr>
<tr>
<td>ldinfo_tls</td>
<td>Thread-local storage region number.</td>
</tr>
</tbody>
</table>

In addition, the ldinfo_filename field has a different interpretation in the __ld_xinfo structure. The value of this field in the __ld_xinfo structure is an offset from the beginning of the structure to a null-terminated path name followed by null-terminated member name.
The `ldr.h` header declares the following functions:

```c
int kmod_load(caddr_t path, uint flags, caddr_t libpath, mid_t *kmid);
int kmod_unload(mid_t kmid, uint flags);
void (*kmod_entrypt(mid_t kmid, uint flags))();
int ld_info(int __flags, pid_t __pid, void *__buffer, unsigned int __length);
__LOAD_T (*load(char *__filenameparm, uint __flags, char *__libpathparm);
int loadbind(int __lflags, void *__exporter, void *__importer);
int unload(void *__function);
int loadquery(int __lflags, void *__exporter, void *__importer);
__handler_t *__lazySetErrorHandler(__handler_t *fp);
```

**Related Information**

The `load`, `loadbind`, `loadquery`, and `unload` subroutines.

---

### limits.h File

**Purpose**

Defines implementation limits identified by IEEE POSIX 1003.

**Description**

The `limits.h` file contains definitions required by the ANSI X3.159-198x Programming Language C Standard and the Institute of Electrical and Electronics Engineers (IEEE) P1003.1 Portable Operating System Interface for Computer Environments (POSIX) standard.

The constants required by the ANSI C Standard describe the sizes of basic data types, as follows:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAR_BIT</td>
<td>8</td>
<td>Number of bits in a variable of type char</td>
</tr>
<tr>
<td>CHAR_MAX</td>
<td>255</td>
<td>Maximum value of a variable of type char</td>
</tr>
<tr>
<td>CHAR_MIN</td>
<td>0</td>
<td>Minimum value of a variable of type char</td>
</tr>
<tr>
<td>INT_MAX</td>
<td>2,147,483,647</td>
<td>Maximum value of a variable of type int</td>
</tr>
<tr>
<td>INT_MIN</td>
<td>-2,147,483,648</td>
<td>Minimum value of a variable of type int</td>
</tr>
<tr>
<td>LONG_MAX</td>
<td>2,147,483,647</td>
<td>Maximum value of a variable of type long</td>
</tr>
<tr>
<td>LONG_MIN</td>
<td>-2,147,483,648</td>
<td>Maximum value of a variable of type long</td>
</tr>
<tr>
<td>SCHAR_MAX</td>
<td>127</td>
<td>Maximum value of a variable of type signed char</td>
</tr>
<tr>
<td>SCHAR_MIN</td>
<td>-128</td>
<td>Minimum value of a variable of type signed char</td>
</tr>
<tr>
<td>SHRT_MAX</td>
<td>32,767</td>
<td>Maximum value of a variable of type short</td>
</tr>
<tr>
<td>SHRT_MIN</td>
<td>-32,768</td>
<td>Minimum value of a variable of type short</td>
</tr>
<tr>
<td>UCHAR_MAX</td>
<td>255</td>
<td>Maximum value of a variable of type unsigned char</td>
</tr>
<tr>
<td>UINT_MAX</td>
<td>4,294,967,295</td>
<td>Maximum value of a variable of type unsigned int</td>
</tr>
<tr>
<td>ULONG_MAX</td>
<td>4,294,967,295</td>
<td>Maximum value of a variable of type unsigned long</td>
</tr>
<tr>
<td>USHRT_MAX</td>
<td>65,535</td>
<td>Maximum value of a variable of type unsigned short</td>
</tr>
</tbody>
</table>

**Run-Time Invariant Values**

The first set of values required by POSIX, run-time invariant values, are simple constants determined by basic operating system data-structure sizes.
### Run-Time Invariant Values (Possibly Indeterminate)

The second set of run-time invariant values required by POSIX specify values that might vary, especially due to system load, but that can be attained on a lightly loaded system.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAX_INPUT</td>
<td>512</td>
<td>No fewer than the number of bytes specified by the MAX_INPUT symbol are allowed in a terminal input queue.</td>
</tr>
<tr>
<td>NGROUPS_MAX</td>
<td>64</td>
<td>Maximum size of the concurrent group list.</td>
</tr>
<tr>
<td>PASS_MAX</td>
<td>32</td>
<td>Maximum number of bytes in a password (not including the null terminator). Only eight characters of password information are significant.</td>
</tr>
<tr>
<td>PID_MAX</td>
<td>INT_MAX</td>
<td>Maximum value for a processID.</td>
</tr>
<tr>
<td>UID_MAX</td>
<td>ULONG_MAX</td>
<td>Maximum value for a user or group ID.</td>
</tr>
</tbody>
</table>

### Note:

The argument list and environment are allowed to consume all of the user data segment.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARG_MAX</td>
<td>24,576&gt;</td>
<td>Maximum length (in bytes) of arguments for the exec subroutine, including the environment</td>
</tr>
</tbody>
</table>

### Path-Name Variable Values

The third set of values required by POSIX, path-name variable values, represent constraints imposed by the file system on file path names. Further constraints on these values might be imposed by the underlying file-system implementation. Use the `pathconf` or `fpathconf` subroutine to determine any file-implementation characteristics specific to the underlying file system.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME_MAX</td>
<td>Undefined</td>
<td>Maximum number of bytes in a file component name (not including the null terminator)</td>
</tr>
<tr>
<td>PATH_MAX</td>
<td>512</td>
<td>Maximum number of bytes in a path name (not including the null terminator)</td>
</tr>
</tbody>
</table>
Run-Time Increasable Values
The fourth set of values required by POSIX specify values that might be increased at run time. Use the `pathconf` or `fpathconf` subroutine to determine any file-implementation characteristics specific to the underlying file system.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>LINK_MAX</td>
<td>32,767</td>
<td>Maximum value of a file's link count (SHRT_MAX).</td>
</tr>
<tr>
<td>PIPE_BUF</td>
<td>32,768</td>
<td>Maximum number of bytes guaranteed to be written automatically to a pipe.</td>
</tr>
</tbody>
</table>

Related Information
The `values.h` file.

The `exec` subroutine, `pathconf` or `fpathconf` subroutine.

libperfstat.h File

Purpose
Describes the structures and constants used by the `libperfstat` API subroutines.

Syntax
```
#include <libperfstat.h>
```

Description
The `libperfstat.h` file defines the following symbolic constants, types, and structures:

- `IDENTIFIER_LENGTH`: length of strings included in the structures
- `FIRST_CPU`: pseudo-name for the first logical cpu
- `FIRST_DISK`: pseudo-name for the first disk
- `FIRST_DISKADAPTER`: pseudo-name for the first disk adapter
- `FIRST_DISKPATH`: pseudo-name for the first disk path
- `FIRST_NETINTERFACE`: pseudo-name for the first network interface
- `FIRST_PAGINGSPACE`: pseudo-name for the first paging space
- `FIRST_PROTOCOL`: pseudo-name for the first protocol
- `FIRST_NETBUFFER`: pseudo-name for the first network buffer size
- `FLUSH_CPUTOTAL`: mask for `perfstat_partial_reset` to flush cached fields for `perfstat_cpu_total_t`
- `FLUSH_DISK`: mask for `perfstat_partial_reset` to flush cached fields for `perfstat_disk_t`
- `RESET_DISK_MIN_MAX`: mask for `perfstat_partial_reset` to reset disk's minimum and maximum service and wait time statistics
- `FLUSH_DISKADAPTER`: mask for `perfstat_partial_reset` to flush cached fields for `perfstat_diskadapter_t`
- `FLUSH_DISKPATH`: mask for `perfstat_partial_reset` to flush cached fields for `perfstat_diskpath_t`
- `FLUSH_PAGINGSPACE`: mask for `perfstat_partial_reset` to flush cached fields for `perfstat_pagingspace_t`
FLUSH_NETINTERFACE

mask for perfstat_partial_reset to flush cached fields for perfstat_netinterface_t

The perfstat_id_t structure contains the following members:

char name [IDENTIFIER_LENGTH] name of the identifier

Some members of the following structures are retrieved only once and stored in a memory cache by the library. A call to perfstat_reset or perfstat_partial_reset will flush those members. When possible, it is preferable to use perfstat_partial_reset instead of perfstat_reset.

The perfstat_cpu_t structure contains the following members:

char name [IDENTIFIER_LENGTH] logical processor name (cpu0, cpu1, ..)
u_longlong_t user raw number of clock ticks spent in user mode
u_longlong_t sys raw number of clock ticks spent in system mode
u_longlong_t idle raw number of clock ticks spent idle
u_longlong_t wait raw number of clock ticks spent waiting for I/O
_u_longlong_t pswitch number of context switches (changes of currently running process)
u_longlong_t syscall number of system calls executed
u_longlong_t sysread number of read system calls executed
u_longlong_t syswrite number of write system calls executed
u_longlong_t sysfork number of fork system call executed
u_longlong_t syscall number of exec system call executed
u_longlong_t readch number of characters transferred with read system call
u_longlong_t writech number of characters transferred with write system call
u_longlong_t bread number of block reads
u_longlong_t bwrite number of block writes
u_longlong_t lread number of logical read requests
u_longlong_t lwrite number of logical write requests
u_longlong_t phread number of physical reads (reads on raw device)
u_longlong_t phwrite number of physical writes (writes on raw device)
u_longlong_t iget number of inode lookups
u_longlong_t namei number of vnode lookup from a path name
u_longlong_t dirblk number of 512-byte block reads by the directory search routine to locate an entry for a file
u_longlong_t msg number of IPC message operations
u_longlong_t sema number of IPC semaphore operations
u_longlong_t minfaults number of page faults with no I/O
u_longlong_t majfaults number of page faults with disk I/O
u_longlong_t puser raw number of physical processor tics in user mode
u_longlong_t psys raw number of physical processor tics in system mode
u_longlong_t pidle raw number of physical processor tics idle
u_longlong_t pwait raw number of physical processor tics waiting for I/O
The `perfstat_cpu_total_t` structure contains the following members:

- `int ncpus`: number of active logical processors
- `int ncpus_cfg`: number of configured processors
- `char description [IDENTIFIER_LENGTH]`: processor description (type/official name)
num_longlong_t processorHZ  
  processor speed in Hz
num_longlong_t user  
  raw total number of clock ticks spent in user mode
num_longlong_t sys  
  raw total number of clock ticks spent in system mode
num_longlong_t idle  
  raw total number of clock ticks spent idle
num_longlong_t wait  
  raw total number of clock ticks spent waiting for I/O
num_longlong_t pswitch  
  number of process switches (change in currently running process)
num_longlong_t syscall  
  number of syscalls executed
num_longlong_t sysread  
  number of read system calls executed
num_longlong_t syswrite  
  number of write system calls executed
num_longlong_t sysfork  
  number of forks system calls executed
num_longlong_t sysexec  
  number of execs system calls executed
num_longlong_t readch  
  number of characters transferred with read system call
num_longlong_t writech  
  number of characters transferred with write system call
num_longlong_t devintrs  
  number of device interrupts
num_longlong_t softintrs  
  number of software interrupts

int lbolt  
  number of ticks since last reboot

t ime_t loadavg[3]  
  (1<<SBITS) times the average number of runnables processes during the
  last 1, 5 and 15 minutes. To calculate the load average, divide the
  numbers by (1<<SBITS). SBITS is defined in <sys/proc.h>.

num_longlong_t runque  
  length of the run queue (processes ready)
num_longlong_t swpque  
  length of the swap queue (processes waiting to be paged in)
num_longlong_t bwrite  
  number of blocks read
num_longlong_t bwrite  
  number of blocks written
num_longlong_t ilread  
  number of logical read requests
num_longlong_t ilwrite  
  number of logical write requests
num_longlong_t phread  
  number of physical reads (reads on raw devices)
num_longlong_t phwrite  
  number of physical writes (writes on raw devices)
num_longlong_t runocc  
  updated whenever runque is updated, i.e. the runqueue is occupied.
  This can be used to compute the simple average of ready processes
num_longlong_t swpocc  
  updated whenever swpque is updated. i.e. the swpqueue is occupied.
  This can be used to compute the simple average processes waiting to be
  paged in

num_longlong_t iget  
  number of inode lookups
num_longlong_t namei  
  number of vnode lookup from a path name
num_longlong_t dirblk  
  number of 512-byte block reads by the directory search routine to
  locate an entry for a file
num_longlong_t msg  
  number of IPC message operations
num_longlong_t sema  
  number of IPC semaphore operations
num_longlong_t rcvint  
  number of tty receive interrupts
num_longlong_t xmtint  
  number of tty transmit interrupts
num_longlong_t mdint  
  number of modem interrupts
num_longlong_t tty_rawin  
  number of raw input characters
num_longlong_t tty_canin  
  number of canonical input characters (always zero)
num_longlong_t tty_rawout  
  number of raw output characters
u_longlong_t ksched
number of kernel processes created

u_longlong_t koverf
number of kernel process creation attempts where:
  • the user has forked to their maximum limit
  • the configuration limit of processes has been reached

u_longlong_t kexit
number of kernel processes that became zombies

u_longlong_t rbread
number of remote read requests

u_longlong_t rbread
number of remote read requests

u_longlong_t rbwrt
number of remote writes

u_longlong_t rcwrt
number of cached remote writes

u_longlong_t traps
number of traps

int ncpus_high
index of highest processor online

u_longlong_t puser
raw number of physical processor tics in user mode

u_longlong_t psys
raw number of physical processor tics in system mode

u_longlong_t pidle raw
number of physical processor tics idle

u_longlong_t pwait raw
number of physical processor tics waiting for I/O

u_longlong_t decrintrs
number of decremter interrupts

u_longlong_t mpcrintrs
number of mpc's received interrupts

u_longlong_t mpcsintrs
number of mpc's sent interrupts

u_longlong_t phantintrs
number of phantom interrupts

u_longlong_t idle_donated_purr
number of idle cycles donated by a dedicated partition enabled for donation

u_longlong_t idle_donated_spurr
number of idle spurr cycles donated by a dedicated partition enabled for donation

u_longlong_t busy_donated_purr
number of busy cycles donated by a dedicated partition enabled for donation

u_longlong_t busy_donated_spurr
number of busy spurr cycles donated by a dedicated partition enabled for donation

u_longlong_t idle_stolen_purr
number of idle cycles stolen by the hypervisor from a dedicated partition

u_longlong_t idle_stolen_spurr
number of idle spurr cycles stolen by the hypervisor from a dedicated partition

u_longlong_t busy_stolen_purr
number of busy cycles stolen by the hypervisor from a dedicated partition

u_longlong_t busy_stolen_spurr
number of busy spurr cycles stolen by the hypervisor from a dedicated partition

The description and speed members of the perfstat_cpu_total_t structure are stored in the information cache. They can be flushed by making one of the following calls:
• perfstat_reset()
• perfstat_partial_reset(NULL,FLUSH_CPUTOTAL)

The perfstat_partition_total_t structure contains the following members:

char name[IDENTIFIER_LENGTH] name[IDENTIFIER_LENGTH]
perfsat_partition_type_t type set of bits describing partition type
int lpar_id logical partition identifier
int group_id identifier of the LPAR group this partition is a member of
int pool_id identifier of the shared pool of physical processors this partition is a member of
int online_cpus number of virtual CPUs currently online on the partition
int max_cpus maximum number of virtual CPUs this partition can ever have
int min_cpus minimum number of virtual CPUs this partition must have
u_longlong_t online_memory amount of memory currently online
u_longlong_t max_memory maximum amount of memory this partition can ever have
u_longlong_t min_memory minimum amount of memory this partition must have
int entitled_proc_capacity number of processor units this partition is entitled to receive
int max_proc_capacity maximum number of processor units this partition can ever have
int min_proc_capacity minimum number of processor units this partition must have
int proc_capacity_increment increment value to the entitled capacity
int unalloc_proc_capacity number of processor units currently unallocated in the shared processor pool this partition belongs to
int var_proc_capacity_weight partition priority weight to receive extra capacity
int unalloc_var_proc_capacity_weight unallocated in the shared processor pool this partition belongs to
int online_phys_cpus_sys number of physical CPUs currently active in the system containing this partition
int max_phys_cpus_sys maximum possible number of physical CPUs in the system containing this partition
int phys_cpus_pool number of the physical CPUs currently in the shared processor pool this partition belong to
u_longlong_t puser raw number of physical processor tics in user mode
u_longlong_t psys raw number of physical processor tics in sytem mode
u_longlong_t pidle raw number of physical processor tics idle
u_longlong_t pwait raw number of physical processor tics waiting for I/O
u_longlong_t pool_idle_time number of clock tics a processor in the shared pool was idle
u_longlong_t phantintrs number of phantom interrupts
u_longlong_t invol_virt_cswitch number involuntary virtual CPU context switches
u_longlong_t vol_virt_cswitch number voluntary virtual CPU context switches
u_longlong_t timebase_last most recent cpu time base timestamp
u_longlong_t reserved_pages number of 16GB pages; cannot participate in DR operations
u_longlong_t reservedpagesize 16GB pagesize; cannot participate in DR operations
u_longlong_t idle_donated_purr number of idle cycles donated by a dedicated partition enabled for donation
u_longlong_t idle_donated_spurr number of idle spurr cycles donated by a dedicated partition enabled for donation
u_longlong_t busy_donated_purr number of busy cycles donated by a dedicated partition enabled for donation
u_longlong_t busy_donated_spurr number of busy spurr cycles donated by a dedicated partition enabled for donation
u_longlong_t idle_stolen_purr number of idle cycles stolen by the hypervisor from a dedicated partition
u_longlong_t idle_stolen_spurr
number of idle spurr cycles stolen by the hypervisor from a dedicated partition

u_longlong_t busy_stolen_spurr
number of busy cycles stolen by the hypervisor from a dedicated partition

u_longlong_t busy_stolen_spurr
number of busy spurr cycles stolen by the hypervisor from a dedicated partition

The perfstat_partition_type_t structure contains the following members:

unsigned:1 smt_capable
OS supports SMT mode

unsigned:1 smt_enabled
SMT mode is on

unsigned:1 lpar_capable
OS supports logical partitioning

unsigned:1 lpar_enabled
logical partitioning is on

unsigned:1 shared_capable
OS supports shared processor LPAR

unsigned:1 shared_enabled
partition runs in shared mode

unsigned:1 dlpar_capable
OS supports dynamic LPAR

unsigned:1 capped
partition is capped

unsigned:1 kernel_is_64
kernel is 64 bits

unsigned:1 pool_util_authority
pool utilization data is available

unsigned:1 donate_capable
Partition is capable of donating cycles.

unsigned:1 donate_enabled
Partition is enabled for donating cycles.

The perfstat_disk_t structure contains the following members:

char name[IDENTIFIER_LENGTH]
name of the disk

char description[IDENTIFIER_LENGTH]
disk description (from ODM)

char vgname[IDENTIFIER_LENGTH]
volume group name (from ODM)

u_longlong_t size
size of the disk (in MB)

u_longlong_t free
free portion of the disk (in MB)

u_longlong_t bsize
disk block size(in bytes)

u_longlong_t __rxfers
number of transfers from disk

u_longlong_t xfers
number of transfers to/from disk

u_longlong_t wbkg
number of blocks written to disk

u_longlong_t rbkg
number of blocks read from disk

u_longlong_t time
amount of time disk is active

char adapter[IDENTIFIER_LENGTH]
disk adapter name

uint paths_count
number of paths defined to the disk

u_longlong_t qdepth
instantaneous "service" queue depth (number of requests sent to disk and not completed yet)

u_longlong_t q_full
"service" queue full occurrence count (number of times the disk is not accepting any more request)

u_longlong_t q_sampled
accumulated sampled "service" queue depth

u_longlong_t rserv
read service time

u_longlong_t rtimeout
number of read request timeouts

u_longlong_t rfailed
number of failed read requests

u_longlong_t min_rserv
min read service time

u_longlong_t max_rserv
max read service time

u_longlong_t wserv
write service time

u_longlong_t wtimeout
number of write request timeouts

su_longlong_t wfailed
number of failed write requests

u_longlong_t min_wserv
min write service time

u_longlong_t max_wserv
max write service time

u_longlong_t wq_depth
instantaneous wait queue depth (number of requests waiting to be sent to disk)

u_longlong_t wq_sampled
accumulated sampled wq_depth
The size, free, vname, adapter and description members of the `perfstat_disk_t` structure are stored in the information cache. They can be flushed by making one of the following calls:

- `perfstat_reset()`
- `perfstat_partial_reset(NULL, FLUSH_DISK)`
- `perfstat_partial_reset("disk name", FLUSH_DISK)`
- `perfstat_partial_reset(NULL, FLUSH_DISKADAPTER)`
- `perfstat_partial_reset("adapter name of this disk", FLUSH_DISKADAPTER)`

The `perfstat_disk_total_t` structure contains the following members:

- `int number` total number of disks
- `u_longlong_t size` total size of all disks (in MB)
- `u_longlong_t free` free portion of all disks (in MB)
- `u_longlong_t __rxfers` total number of transfers from disk
- `u_longlong_t xfers` total number of transfers to/from disk
- `u_longlong_t wblks` 512 bytes blocks written to all disks
- `u_longlong_t rblks` 512 bytes blocks read from all disks
- `u_longlong_t time` amount of time disks are active

The size and free members of the `perfstat_disk_total_t` structure are stored in the information cache. They can be flushed by making one of the following calls:

- `perfstat_reset()`
- `perfstat_partial_reset(NULL, FLUSH_DISK)`
- `perfstat_partial_reset("disk name", FLUSH_DISK)`
- `perfstat_partial_reset(NULL, FLUSH_DISKADAPTER)`
- `perfstat_partial_reset("adapter name of this disk", FLUSH_DISKADAPTER)`

The `perfstat_diskadapter_t` structure contains the following members:

- `char name[IDENTIFIER_LENGTH]` name of the adapter (from ODM)
- `char description[IDENTIFIER_LENGTH]` adapter description (from ODM)
- `int number` number of disks connected to adapter
- `u_longlong_t size` total size of all disks (in MB)
- `u_longlong_t free` free portion of all disks (in MB)
- `u_longlong_t __rxfers` total number of reads via adapter
- `u_longlong_t xfers` total number of transfers to/from disk
- `u_longlong_t wblks` 512 bytes blocks written via adapter
- `u_longlong_t rblks` 512 bytes blocks read via adapter
- `u_longlong_t time` amount of time disks are active

The list of the disk adapters and the size, free, and description members of the `perfstat_diskadapter_t` structure are stored in the information cache. They can be flushed by making one of the following calls:

- `perfstat_reset()`
- `perfstat_partial_reset(NULL, FLUSH_DISKADAPTER)`
- `perfstat_partial_reset("diskadapter name", FLUSH_DISKADAPTER)`
- `perfstat_partial_reset(NULL, FLUSH_DISK)`
- `perfstat_partial_reset("disk linked to this adapter", FLUSH_DISK)`

The `perfstat_diskpath_t` structure contains the following members:

- `char name[IDENTIFIER_LENGTH]` name of the path
- `u_longlong_t __rxfers` number of reads via the path
- `u_longlong_t xfers` total number of transfers via the path
- `u_longlong_t wblks` 512 bytes blocks written via the path
- `u_longlong_t rblks` 512 bytes blocks read via the path
The adapter member of the perfstat_diskpath_t structure is stored in the information cache. It can be flushed by making one of the following calls:

- `perfstat_reset()`
- `perfstat_partial_reset(NULL, FLUSH_DISKPATH)`
- `perfstat_partial_reset("diskpath name", FLUSH_DISKPATH)`
- `perfstat_partial_reset(NULL, FLUSH_DISKADAPTER)`
- `perfstat_partial_reset("diskadapter name of this diskpath", FLUSH_DISKADAPTER)`
- `perfstat_partial_reset(NULL, FLUSH_DISK)`
- `perfstat_partial_reset("disk pointed by this path", FLUSH_DISK)`

The perfstat_memory_total_t structure contains the following members:

- `u_longlong_t virt_total` total virtual memory (in 4KB pages)
- `u_longlong_t real_total` total real memory (in 4KB pages)
- `u_longlong_t real_free` free real memory (in 4KB pages)
- `u_longlong_t real_pinned` real memory which is pinned (in 4KB pages)
- `u_longlong_t real_inuse` real memory which is in use (in 4KB pages)
- `u_longlong_t pgbad` number of bad pages
- `u_longlong_t pgexct` number of page faults
- `u_longlong_t pgins` number of pages pagd in
- `u_longlong_t pgouts` number of pages pagd out
- `u_longlong_t pgspins` number of page ins from paging space
- `u_longlong_t pgspouts` number of page outs from paging space
- `u_longlong_t scans` number of page scans by clock
- `u_longlong_t cycles` number of page replacement cycles
- `u_longlong_t pgsteals` number of page steals
- `u_longlong_t numperm` number of frames used for files (in 4KB pages)
- `u_longlong_t pgsp_total` total paging space (in 4KB pages)
- `u_longlong_t pgsp_free` free paging space (in 4KB pages)
- `u_longlong_t pgsp_rsvd` reserved paging space (in 4KB pages)
- `u_longlong_t real_system` real memory used by system segments (in 4KB pages). This is the sum of all the used pages in segment marked for system usage. Since segment classifications are not always guaranteed to be accurate, this number is only an approximation.
- `u_longlong_t real_user` real memory used by non-system segments (in 4KB pages). This is the sum of all pages used in segments not marked for system usage. Since segment classifications are not always guaranteed to be accurate, this number is only an approximation.
u_longlong_t real_process  real memory used by process segments (in 4KB pages).
Since real_system is an approximation, this number is as well.

u_longlong_t virt_active  Active virtual pages (avm column in vmstat output). Virtual pages are
considered active if they have been accessed.

The perfstat_netinterface_t structure contains the following members:

<table>
<thead>
<tr>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>char name[IDENTIFIER_LENGTH]</td>
<td>name of the interface</td>
</tr>
<tr>
<td>char description[IDENTIFIER_LENGTH]</td>
<td>interface description (from ODM, similar to lscfg output)</td>
</tr>
<tr>
<td>uchar type</td>
<td>ethernet, tokenring, etc. interpretation can be done using /usr/include/net/if_types.h</td>
</tr>
<tr>
<td>u_longlong_t mtu</td>
<td>network frame size</td>
</tr>
<tr>
<td>u_longlong_t ipackets</td>
<td>number of packets received on interface</td>
</tr>
<tr>
<td>u_longlong_t ibytes</td>
<td>number of bytes received on interface</td>
</tr>
<tr>
<td>u_longlong_t terrors</td>
<td>number of input errors on interface</td>
</tr>
<tr>
<td>u_longlong_t opackets</td>
<td>number of packets sent on interface</td>
</tr>
<tr>
<td>u_longlong_t obytes</td>
<td>number of bytes sent on interface</td>
</tr>
<tr>
<td>u_longlong_t terrors</td>
<td>number of output errors on interface</td>
</tr>
<tr>
<td>u_longlong_t collisions</td>
<td>number of collisions on csma interface</td>
</tr>
<tr>
<td>u_longlong_t bitrate</td>
<td>adapter rating in bit per second</td>
</tr>
</tbody>
</table>

The description member of the perfstat_netinterface_t structure is stored in the information cache. It can be flushed by making one of the following calls:

- perfstat_reset()
- perfstat_partial_reset(NULL, FLUSH_NETINTERFACE)
- perfstat_partial_reset(“netinterface name”, FLUSH_NETINTERFACE)

The perfstat_netinterface_total_t structure contains the following members:

<table>
<thead>
<tr>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int number</td>
<td>number of network interfaces</td>
</tr>
<tr>
<td>u_longlong_t ipackets</td>
<td>number of packets received on interface</td>
</tr>
<tr>
<td>u_longlong_t ibytes</td>
<td>number of bytes received on interface</td>
</tr>
<tr>
<td>u_longlong_t terrors</td>
<td>number of input errors on interface</td>
</tr>
<tr>
<td>u_longlong_t opackets</td>
<td>number of packets sent on interface</td>
</tr>
<tr>
<td>u_longlong_t obytes</td>
<td>number of bytes sent on interface</td>
</tr>
<tr>
<td>u_longlong_t terrors</td>
<td>number of output errors on interface</td>
</tr>
<tr>
<td>u_longlong_t collisions</td>
<td>number of collisions on csma interface</td>
</tr>
</tbody>
</table>

The perfstat_pagingspace_t structure contains the following members:

<table>
<thead>
<tr>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>char name[IDENTIFIER_LENGTH]</td>
<td>paging space name</td>
</tr>
<tr>
<td>char type</td>
<td>type of paging device. Possible values are:</td>
</tr>
<tr>
<td></td>
<td>LV_PAGING  logical volume</td>
</tr>
<tr>
<td></td>
<td>NFS_PAGING  NFS file</td>
</tr>
</tbody>
</table>

The nfs_paging union has the following fields:

<table>
<thead>
<tr>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>char nfs_paging.hostname[IDENTIFIER_LENGTH]</td>
<td>host name of paging server</td>
</tr>
<tr>
<td>char nfs_paging.filename[IDENTIFIER_LENGTH]</td>
<td>swap file name on server</td>
</tr>
</tbody>
</table>

The lv_paging union has the following fields:

<table>
<thead>
<tr>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>char lv_paging.vgname[IDENTIFIER_LENGTH]</td>
<td>volume group name</td>
</tr>
<tr>
<td>longlong_t lp_size</td>
<td>size in number of logical partitions</td>
</tr>
<tr>
<td>longlong_t mb_size</td>
<td>size in megabytes</td>
</tr>
<tr>
<td>longlong_t mb_used</td>
<td>portion used in megabytes</td>
</tr>
<tr>
<td>longlong_t io_pending</td>
<td>number of pending I/O</td>
</tr>
<tr>
<td>char active</td>
<td>indicates if active (1 if so, 0 if not)</td>
</tr>
<tr>
<td>char automatic</td>
<td>indicates if automatic (1 if so, 0 if not)</td>
</tr>
</tbody>
</table>
The list of the paging spaces and the automatic, type, lpsize, mbsize, hostname, filename, and vgname members of the `perfstat_pagingspace_t` structure are stored in the information cache. They can be flushed by making one of the following calls:

- `perfstat_reset()`
- `perfstat_partial_reset(NULL, FLUSH_PAGINGSPACE)`
- `perfstat_partial_reset("paging space name", FLUSH_PAGINGSPACE)`

The `perfstat_netbuffer_t` structure contains the following members:

- `char name[IDENTIFIER_LENGTH]` size in ascii, always power of 2 (ex: "32", "64", "128")
- `u_longlong_t inuse` number of buffer currently allocated
- `u_longlong_t calls` number of buffer allocations since last reset
- `u_longlong_t delayed` number of delayed allocations
- `u_longlong_t free` number of free calls
- `u_longlong_t failed` number of failed allocations
- `u_longlong_t highwatermark` high threshold for number of buffer allocated
- `u_longlong_t freed` number of buffers freed

The `perfstat_protool_t` structure contains the following members:

- `char name[IDENTIFIER_LENGTH]` ip, ipv6, icmp, icmpv6, udp, tcp, rpc, nfs, nfsv2, nfsv8

The ip union contains the following fields:

- `u_longlong_t ip.ipackets` number of input packets
- `u_longlong_t ip.ierrors` number of input errors
- `u_longlong_t ip.iqueueoverflow` number of input queue overflows
- `u_longlong_t ip.opackets` number of output packets
- `u_longlong_t ip.oerrors` number of output errors

The ipv6 union contains the following fields:

- `_longlong_t ipv6.ipackets` number of input packets
- `u_longlong_t ipv6.ierrors` number of input errors
- `u_longlong_t ipv6.iqueueoverflow` number of input queue overflows
- `u_longlong_t ipv6.opackets` number of output packets
- `u_longlong_t ipv6.oerrors` number of output errors

The icmp union contains the following fields:

- `u_longlong_t icmp.received` number of packets received
- `u_longlong_t icmp.sent` number of packets sent
- `u_longlong_t icmp.errors` number of errors

The icmpv6 union contains the following fields:

- `u_longlong_t icmpv6.received` number of packets received
- `u_longlong_t icmpv6.sent` number of packets sent
- `u_longlong_t icmpv6.errors` number of errors

The udp union contains the following fields:

- `u_longlong_t udp.ipackets` number of input packets
- `u_longlong_t udp.ierrors` number of input errors
- `u_longlong_t udp.opackets` number of output packets
- `u_longlong_t udp.no_socket` number of packets dropped due to no socket

The tcp union contains the following fields:

- `u_longlong_t tcp.ipackets` number of input packets
- `u_longlong_t tcp.ierrors` number of input errors
- `u_longlong_t tcp.opackets` number of output packets
- `u_longlong_t tcp.initiated` number of connections initiated
The rpc union contains the following fields:

- `u_longlong_t rpc.client.stream.calls`: total NFS client RPC connection-oriented calls
- `u_longlong_t rpc.client.stream.badcalls`: rejected NFS client RPC calls
- `u_longlong_t rpc.client.stream.badxids`: bad NFS client RPC call responses
- `u_longlong_t rpc.client.stream.timeout`: timeout NFS client RPC calls with no reply
- `u_longlong_t rpc.client.stream.newcreds`: total NFS client RPC authentication refreshes
- `u_longlong_t rpc.client.stream.badverfs`: total NFS client RPC bad verifier in response
- `u_longlong_t rpc.client.stream.timers`: NFS client RPC timeout greater than timeout value
- `u_longlong_t rpc.client.stream.nomem`: NFS client RPC calls memory allocation failure
- `u_longlong_t rpc.client.stream.cantconn`: failed NFS client RPC calls
- `u_longlong_t rpc.client.stream.interupts`: NFS client RPC calls fail due to interrupt
- `u_longlong_t rpc.client.dgram.calls`: total NFS client RPC connectionless calls
- `u_longlong_t rpc.client.dgram.badcalls`: rejected NFS client RPC calls
- `u_longlong_t rpc.client.dgram.retrans`: retransmitted NFS client RPC calls
- `u_longlong_t rpc.client.dgram.badxids`: bad NFS client RPC call responses
- `u_longlong_t rpc.client.dgram.timeout`: timed out NFS client RPC calls with no reply
- `u_longlong_t rpc.client.dgram.newcreds`: total NFS client RPC authentication refreshes
- `u_longlong_t rpc.client.dgram.badverfs`: total NFS client RPC bad verifier in response
- `u_longlong_t rpc.client.dgram.timers`: NFS client RPC timeout greater than timeout value
- `u_longlong_t rpc.client.dgram.nomem`: NFS client RPC calls memory allocation failure
- `u_longlong_t rpc.client.dgram.cantconnect`: NFS client RPC calls not sent
- `u_longlong_t rpc.server.stream.calls`: total NFS server RPC connection-oriented requests
- `u_longlong_t rpc.server.stream.badcalls`: rejected NFS server RPC requests
- `u_longlong_t rpc.server.stream.nullrecv`: NFS server RPC calls failed due to unavailable packet
- `u_longlong_t rpc.server.stream.badlen`: NFS server RPC requests failed due to bad length
- `u_longlong_t rpc.server.stream.xdrcall`: NFS server RPC requests failed due to bad header
- `u_longlong_t rpc.server.stream.dupchecks`: NFS server RPC calls found in request cache
- `u_longlong_t rpc.server.dgram.calls`: total NFS server RPC connectionless requests
- `u_longlong_t rpc.server.dgram.badcalls`: rejected NFS server RPC requests
- `u_longlong_t rpc.server.dgram.nullrecv`: NFS server RPC calls failed due to unavailable packet
- `u_longlong_t rpc.server.dgram.badlen`: NFS server RPC requests failed due to bad length
- `u_longlong_t rpc.server.dgram.xdrcall`: NFS server RPC requests failed due to bad header
- `u_longlong_t rpc.server.dgram.dupchecks`: NFS server RPC calls found in request cache
- `u_longlong_t rpc.server.dgram.dupreqs`: total NFS server RPC call duplicates

The nfs union contains the following fields:

- `u_longlong_t nfs.client.calls`: total NFS client requests
- `u_longlong_t nfs.client.badcalls`: total NFS client failed calls
- `u_longlong_t nfs.client.clget`: total number of client nfs clgets
- `u_longlong_t nfs.client.cltomany`: total number of client nfs cltomany
- `u_longlong_t nfs.server.calls`: total NFS server requests
- `u_longlong_t nfs.server.badcalls`: total NFS server failed calls
- `u_longlong_t nfs.server.public_v2`: total number of nfs version 2 server calls
- `u_longlong_t nfs.server.public_v3`: total number of nfs version 3 server calls

The nfsv2 union contains the following fields:

- `u_longlong_t nfsv2.client.calls`: NFS V2 client requests
- `u_longlong_t nfsv2.client.null`: NFS V2 client null requests
- `u_longlong_t nfsv2.client.getattr`: NFS V2 client getattr requests
- `u_longlong_t nfsv2.client.setattr`: NFS V2 client setattr requests
- `u_longlong_t nfsv2.client.root`: NFS V2 client root requests
- `u_longlong_t nfsv2.client.lookup`: NFS V2 client file name lookup requests
- `u_longlong_t nfsv2.client.readlink`: NFS V2 client readlink requests
The nfsv3 union contains the following fields:

- `u_longlong_t nfsv3.client.calls`: NFS V3 client calls
- `u_longlong_t nfsv3.client.null`: NFS V3 client null requests
- `u_longlong_t nfsv3.client.getattr`: NFS V3 client getattr requests
- `u_longlong_t nfsv3.client.setattr`: NFS V3 client setattr requests
- `u_longlong_t nfsv3.client.lookup`: NFS V3 client file name lookup requests
- `u_longlong_t nfsv3.client.access`: NFS V3 client access requests
- `u_longlong_t nfsv3.client.readlink`: NFS V3 client readlink requests
- `u_longlong_t nfsv3.client.read`: NFS V3 client read requests
- `u_longlong_t nfsv3.client.write`: NFS V3 client write requests
- `u_longlong_t nfsv3.client.create`: NFS V3 client file creation requests
- `u_longlong_t nfsv3.client.mknod`: NFS V3 client mknod creation requests
- `u_longlong_t nfsv3.client.remove`: NFS V3 client file removal requests
- `u_longlong_t nfsv3.client.rename`: NFS V3 client file rename requests
- `u_longlong_t nfsv3.client.mkdir`: NFS V3 client directory creation requests
- `u_longlong_t nfsv3.client.rmtree`: NFS V3 client directory removal requests
- `u_longlong_t nfsv3.client.readdirplus`: NFS V3 client read-directory plus requests
- `u_longlong_t nfsv3.client.statfs`: NFS V3 client statfs requests
- `u_longlong_t nfsv3.client.lstat`: NFS V3 client lstat requests
- `u_longlong_t nfsv3.client.l.sb`: NFS V3 client lstat sb
- `u_longlong_t nfsv3.client.pathconf`: NFS V3 client pathconf requests
- `u_longlong_t nfsv3.client.mount`: NFS V3 client mount requests
- `u_longlong_t nfsv3.client.umount`: NFS V3 client umount requests
- `u_longlong_t nfsv3.client.commit`: NFS V3 client commit requests
- `u_longlong_t nfsv3.server.calls`: NFS V3 server requests
- `u_longlong_t nfsv3.server.null`: NFS V3 server null requests
- `u_longlong_t nfsv3.server.getattr`: NFS V3 server getattr requests
- `u_longlong_t nfsv3.server.setattr`: NFS V3 server setattr requests
- `u_longlong_t nfsv3.server.lookup`: NFS V3 server file name lookup requests
- `u_longlong_t nfsv3.server.access`: NFS V3 server access requests
- `u_longlong_t nfsv3.server.read`: NFS V3 server read requests
- `u_longlong_t nfsv3.server.write`: NFS V3 server write requests
- `u_longlong_t nfsv3.server.create`: NFS V3 server file creation requests
- `u_longlong_t nfsv3.server.remove`: NFS V3 server file removal requests
- `u_longlong_t nfsv3.server.rename`: NFS V3 server file rename requests
- `u_longlong_t nfsv3.server.mkdir`: NFS V3 server directory creation requests
- `u_longlong_t nfsv3.server.rmtree`: NFS V3 server directory removal requests
- `u_longlong_t nfsv3.server.readdir`: NFS V3 server read requests
- `u_longlong_t nfsv3.server.readdirplus`: NFS V3 server read-directory plus requests
- `u_longlong_t nfsv3.server.statfs`: NFS V3 server statfs requests
- `u_longlong_t nfsv3.server.lstat`: NFS V3 server lstat requests
- `u_longlong_t nfsv3.server.l.sb`: NFS V3 server lstat sb
- `u_longlong_t nfsv3.server.pathconf`: NFS V3 server pathconf requests
- `u_longlong_t nfsv3.server.mount`: NFS V3 server mount requests
- `u_longlong_t nfsv3.server.umount`: NFS V3 server umount requests
- `u_longlong_t nfsv3.server.commit`: NFS V3 server commit requests
The following are declared as functions:

```c
int perfstat_cpu(perfstat_id_t *name,
                 perfstat_cpu_t *userbuff,
                 int sizeof_userbuff,
                 int desired_number)

int perfstat_cpu_total(perfstat_id_t *name,
                        perfstat_cpu_total_t *userbuff,
                        int sizeof_userbuff,
                        int desired_number)

int perfstat_partition_total(perfstat_id_t *name,
                             perfstat_partition_total_t *userbuff,
                             int sizeof_userbuff,
                             int devid_number)

int perfstat_disk(perfstat_id_t *name,
                  perfstat_disk_t *userbuff,
                  int sizeof_userbuff,
                  int desired_number)

int perfstat_disk_total(perfstat_id_t *name,
                         perfstat_disk_total_t *userbuff,
                         int sizeof_userbuff,
                         int desired_number)

int perfstat_diskadapter(perfstat_id_t *name,
                         perfstat_diskadapter_t *userbuff,
                         int sizeof_userbuff,
                         int desired_number)

int perfstat_memory_total(perfstat_id_t *name,
                          perfstat_memory_total_t *userbuff,
                          int sizeof_userbuff,
                          int desired_number)

int perfstat_netinterface(perfstat_id_t *name,
                          perfstat_netinterface_t *userbuff,
                          int sizeof_userbuff,
                          int desired_number)

int perfstat_netinterface_total(perfstat_id_t *name,
                                 perfstat_netinterface_total_t *userbuff,
                                 int sizeof_userbuff,
                                 int desired_number)

int perfstat_pagingspace(perfstat_id_t *name,
                        perfstat_pagingspace_t *userbuff,
                        int sizeof_userbuff,
                        int desired_number)

int perfstat_netbuffer(perfstat_id_t *name,
                       perfstat_netbuffer_t *userbuff,
                       int sizeof_userbuff,
                       int desired_number)

int perfstat_protocol(perfstat_id_t *name,
                      perfstat_protocol_t *userbuff,
                      int sizeof_userbuff,
                      int desired_number)
```
```
int desired_number
int perfstat_partial_reset(char *name,
    u_longlong_t defmask)
void perfstat_reset(void)
```

**Related Information**

The `perfstat_cpu` subroutine, `perfstat_cpu_total` subroutine, `perfstat_memory_total` subroutine, `perfstat_disk` subroutine, `perfstat_disk_total` subroutine, `perfstat_diskadapter` subroutine, `perfstat_diskpath` subroutine, `perfstat_netinterface` subroutine, `perfstat_netinterface_total` subroutine, `perfstat_partition_total` subroutine, `perfstat_partial_reset` subroutine, and `perfstat_reset` subroutines.

---

**math.h File**

**Purpose**

Defines math subroutines and constants.

**Description**

The `/usr/include/math.h` header file contains declarations of all the subroutines in the Math library (`libm.a`) and of various subroutines in the Standard C Library (`libc.a`) that return floating-point values.

Among other things, the `math.h` file defines the following macro, which is used as an error-return value:

```
HUGE_VAL
```

Specifies the maximum value of a double-precision floating-point number: +infinity on machines that support IEEE-754 and `DBL_MAX` otherwise.

If you define the `__MATH__` preprocessor variable before including the `math.h` file, the `math.h` file defines macros that make the names of certain math subroutines appear to the compiler as `__xxxx`. The following names are redefined to have the `__` (double underscore) prefix:

```
exp    sin
asin   log
cos    acos
log10  tan
atan   sqrt
fabs   atan2
```

These special names instruct the C compiler to generate code that avoids the overhead of the Math library subroutines and issues compatible-mode floating-point subroutines directly. The `__MATH__` variable is defined by default.

If `_XOPEN_SOURCE` variable is defined, the following mathematical constants are defined for your convenience. The values are of type double and are accurate to the precision of this type. That is, the machine value is the mathematical value rounded to double precision.

```
M_E     Base of natural logarithms (e)
M_LOG2E Base-2 logarithm of e
M_LOG10E Base-10 logarithm of e
M_LN2   Natural logarithm of 2
M_LN10  Natural logarithm of 10
M_PI    Pi, the ratio of the circumference of a circle to its diameter
```
Related Information

The `values.h` file.

**mode.h File**

**Purpose**
Defines the interpretation of a file mode.

**Description**
This version of the operating system supports a 32-bit mode, which is divided into 3 parts. The 16 most significant bits are reserved by the system. The least significant 16 bits define the type of file (`S_IFMT`) and the permission bits. The 12 permission bits can be changed by using the `chmod` or `chac` subroutine. The file type cannot be changed.

**File-Type Bits**
The file type determines the operations that can be applied to the file (including implicit operations, such as searching a directory or following a symbolic link). The file type is established when the file is created, and cannot be changed. The following file types are supported:

- `S_IFDIR` Defines a directory.
- `S_IFREG` Defines a regular file.
- `S_IFIFO` Defines a pipe.
- `S_IFCHR` Defines a character device.
- `S_IFBLK` Defines a block device.
- `S_IFLNK` Defines a symbolic link.
- `S_IFSOCK` Defines a socket.

The `S_IFMT` format mask constant can be used to mask off a file type from the mode.

**File-Attribute Bits**
The file-attribute bits affect the interpretation of a particular file. With some restrictions, file attributes can be changed by the owner of a file or by a privileged user. The file-attribute bits are:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
</table>

**S_ISUID Bit:**

- `setuid` When a process runs a regular file that has the `S_ISUID` bit set, the effective user ID of the process is set to the owner ID of the file. The `setuid` attribute can be set only by a process on a trusted path. If the file or its access permissions are altered, the `S_ISUID` bit is cleared.
**S_ISGID (S_ENFMT) Bit:**

**setgid**  When a process runs a regular file that has both the S_ISGID bit and the S_IXGRP permission bit set, the effective user ID of the process is set to the group ID of the file. The setgid attribute can be set only by a process on a trusted path. If the owner is establishing this attribute, the group of the file must be the effective group ID or in the supplementary group ID of the process. If the file or its access permissions are altered, the S_ISGID bit is cleared.

**enforced locking**  If a regular file has the S_ISGID bit set and the S_IXGRP permission bit cleared, locks placed on the file with the lockfx subroutine are enforced locks.

**S_IFMPX Bit:**

**multiplexed**  A character device with the S_IFMPX attribute bit set is a multiplexed device. This attribute is established when the device is created.

**S_ISVTX Bit:**

**sticky**  If a directory has the S_SVTX bit set, only the owner of the file or the owner of the directory can remove a file from the directory.

**S_IACL Bit:**

**access control list**  Any file that has the S_IACL bit set can have an extended access control list (ACL). Specifying this bit when setting the mode with the chmod command causes the permission bits information in the mode to be ignored. Extended ACL entries are ignored if this bit is cleared. This bit can be implicitly cleared by the chmod subroutine. The /usr/include/sys/acl.h file defines the format of the ACL.

**S_ITCB Bit:**

**trusted**  Any file that has the S_ITCB bit set is part of the Trusted Computing Base (TCB). Only files in the TCB can acquire privilege on a trusted path. Only files in the TCB are run by the trusted shell (which is invoked with the tsh command). This attribute can be established or cleared only by a process running on the trusted path.

**S_IJRNAL Bit:**

**journaled**  Any file that has the S_IJRNAL bit set is defined as a journaled file. Updates to a journaled file are added to a log atomically. All directories and system files have the journaled attribute, which cannot be reset.

**File-Permission Bits**

The file-permission bits control which processes can perform operations on a file. This includes read, write, and execute bits for the file owner, the file group, and the default. These bits should not be used to set access-control information; the ACL should be used instead. The file-permission bits are:

- **S_IRWXU**  Permits the owner of a file to read, write, and execute the file.
- **S_IRUSR**  Permits the owner of a file to read the file.
- **S_IREAD**  Permits the owner of a file to read the file.
- **S_IWUSR**  Permits the owner of a file to write to the file.
- **S_IWRITE**  Permits the owner of a file to write to the file.
- **S_IXUSR**  Permits the owner of a file to execute the file or to search the file’s directory.
S_IEXEC  Permits the owner of a file to execute the file or to search the file's directory.
S_IRWXG  Permits a file's group to read, write, and execute the file.
S_IRGRP  Permits a file's group to read the file.
S_IWGRP  Permits a file's group to write to the file.
S_IXGRP  Permits a file's group to execute the file or to search the file's directory.
S_IRWIXO Permits others to read, write, and execute the file.
S_IROTH  Permits others to read the file.
S_IWOTH  Permits others to write to the file.
S_IXOTH  Permits others to execute the file or to search the file's directory.

Related Information
The stat.h file, types.h file.
The chmod command, tsh command.

msg.h File

Purpose
Describes the structures that are used by the subroutines that perform message queueing operations.

Syntax
#include <sys/msg.h>

Description
The msg.h file defines the following symbolic constants, types, and structures:

Types:
unsigned int msgqnum_t;
unsigned int msglen_t;

Symbolic Constants:

MSG_NOERROR  no error if big message */
MSG_R        read permission */
MSG_W        write permission */
MSG_RWAIT    a reader is waiting for a message */
MSG_WWAIT    a writer is waiting to send */
MSG_STAT     Number of bytes to copy for IPC_STAT command
MSGXBUFSIZE  the length of everything but mtext[1] and padding
MSG_SYSSPACE for msgsnd() flags
XMSG         for msgrcv() flags

There is one msg queue id data structure for each q in the system. The msqid_ds structure contains the following members:

struct ipc_perm  msg_perm;  operation permission
struct
void *msg_first;  ptr to first message on q
void *msg_last;   ptr to last message on q
unsigned int _msg_cbytes;  current # bytes on q
msgqnum_t  msg_qnum;  # of messages on q
msglen_t   msg_qbytes;  max # of bytes on q
pid_t      msg_lspid;  pid of last msgsnd
pid_t      msg_lrpid;  pid of last msgrcv
The `msg_hdr` struct contains the following members:

```c
struct msg_hdr {
    time_t mtime;          /* time message was sent */
    uid_t muid;            /* author's effective uid */
    gid_t mgid;            /* author's effective gid */
    pid_t mpid;            /* author's process id */
    mtyp_t mtype;          /* message type */
    struct msg *msg_next;  /* ptr to next message on q */
    struct msg hdr msg_attr; /* message attributes */
    unsigned int msg_ts;   /* message text size */
    char *msg_spot;        /* pointer to message text */
};
```

There is one `msg` structure for each message that may be in the system. The `msg` structure contains the following members:

```c
struct msg {
    msg_hdr msg_hdr; /* message attributes */
    char *msg_text;  /* message text */
};
```

The structure `msgbuf` is the user message buffer template for `msgsnd` and `msgrcv` system calls and contains the following members:

```c
struct msgbuf {
    mtyp_t mtype; /* message type */
    char mtext[1]; /* message text */
};
```

The `msgxbuf` structure is the user message buffer template for the `msgxrcv` system call and contains the following members:

```c
struct msgxbuf {
    time_t mtime;          /* time message was sent */
    uid_t muid;            /* author's effective uid */
    gid_t mgid;            /* author's effective gid */
    pid_t mpid;            /* author's process id */
    mtyp_t mtype;          /* Message type */
    char mtext[1];         /* Message text */
};
```

The `msginfo` structure contains the following members:

```c
struct msginfo {
    int msgmax,          /* max message size */
    int msgmnb,          /* max # bytes on queue */
    int msgmni,          /* # of message queue identifiers */
    int msgmmn;          /* max # messages per queue identifier */
};
```

The `time_t`, `size_t`, `off_t`, `mtyp_t`, `pid_t`, and `gid_t` types are as defined in `<sys/types.h>`.

The following are declared as functions:

```c
int msgget(key_t, int);
int msgrcv(int, void *, size_t, long, int);
int msgsnd(int, const void *, size_t, int);
int msgctl(int, int, struct msqid_ds *);
int msgxrcv(int, struct msgxbuf*, int, long, int);
```

In addition, all of the symbols from `<sys/ipc.h>` will be defined when this header is included.

**Related Information**

The `mmap`, `mprotect`, `msync`, and `munmap` subroutines.
mtio.h File

Purpose
Defines the magnetic tape user include file.

Description
The /usr/include/sys/mtio.h file is for use with devices that are driven by the magnetic tape driver. Since Linux has an mtio.h file, the sys/mtio.h file has been added for Linux compatibility. The sys/mtio.h file can be used for ESCON tape drives or for Linux-compatible source code.

Notes:
1. For Linux source-code compatibility (Linux-based source code or definitions), set the following before including the sys/mtio.h file:
   #define __LINUX_SOURCE_COMPAT
2. For an application that uses ESCON tape devices, install the ESCON tape device driver file sets on the build machine to obtain the sys/mtextend.h file. The ESCON tape device driver file sets are devices.pci.esconCH.rte and devices.common.IBM.esconCH.mtd.rte.

File
/usr/include/sys/mtio.h

Related Information
Operating system and device management

param.h File

Purpose
Describes system parameters.

Description
Certain parameters vary for different hardware that uses the operating system. These parameters are defined in the /usr/include/sys/param.h file. The most significant parameters are:

NCARGS Indicates the default number of characters, including terminating null characters, that can be passed using the exec subroutine.
UBSIZE The unit used by the statistics subroutines for returning block sizes of files.

This file also contains macros for manipulating machine-dependent fields.

Programs that are intended to comply with the POSIX standard should include the /usr/include/sys/limits.h file rather than the param.h file.

Related Information
The exec subroutine.

The Header Files Overview defines header files, describes how they are used, and lists several of the header files for which information is provided in this documentation.

The Kernel tunable parameters in Performance management for information on tuning the System Configuration value of the argument list.
pmapi.h File

Purpose
Describes the structures and constants used by the Performance Monitor APIs subroutines.

Syntax
#include <pmapi.h>

Description
The pmapi.h file defines the following symbolic constants, types, and structures:

Symbolic Constants

- **MAX_COUNTERS**: Maximum number of supported counters
- **MIN_THRESH_VALUE**: Minimum threshold value
- **MAX_THRESH_VALUE**: Maximum threshold value
- **COUNT NOTHING**: Specifies to not count event

Constants for event filters

- **PM_VERIFIED**: Specifies events that have been verified
- **PM_UNVERIFIED**: Specifies events that have not been verified
- **PM_CAVEAT**: Specifies events that work with caveats
- **PM_GET_GROUPS**: Not a filter; specifies that supported event groups are to be returned by pm_init.

Constants for Processor Idents

- **PM_CURRENT**: Specifies that the PMAPI is to be initialized for the current processor type.
- **PM_PowerPC604**: Specifies that supported events for the PowerPC604 are to be returned.
- **PM_PowerPC604e**: Specifies that supported events for the PowerPC604e are to be returned.
- **PM_RS64_II**: Specifies that supported events for the RS64-II are to be returned.
- **PM_POWER3**: Specifies that supported events for the POWER3 are to be returned.
- **PM_RS64_III**: Specifies that supported events for the RS64-III are to be returned.
- **PM_POWER3_II**: Specifies that supported events for the POWER3-II are to be returned.
- **PM_POWER4**: Specifies that supported events and/or groups for the POWER4 are to be returned.
- **PM_MPC7450**: Specifies that supported events for the MPC7450 are to be returned.
- **PM_POWER4_II**: Specifies that supported events and/or groups for the POWER4-II are to be returned.
- **PM_POWER5**: Specifies that supported events and/or groups for the POWER5 are to be returned.
- **PM_PowerPC 970**: Specifies that supported events and/or groups for the PowerPC 970 are to be returned.
- **PM_POWER6**: Specifies that supported events and/or groups for the POWER6 are to be returned.
- **PM_MAXPROCTYPE**: Maximum number of processor idents.

Constants for setting mode bits

- **PM_PROCTREE**: Turns process tree counting on
- **PM_COUNT**: Turns counting on immediately
- **PM_USER**: Turns user mode counting on
- **PM_KERNEL**: Turns kernel mode counting on
- **PM_PROCESS**: Creates a process level group

The structure pm_info_t contains the following members:
The structure `pm_events_t` contains the following members:

- `int event_id`: event number
- `char status`: 'v': verified  
  'u': unverified  
  'c': caveat
- `char threshold`: 'y': thresholdable  
  'g': group-only  
  'G': thresholdable group-only
- `char *short_name`: mnemonic name
- `char *long_name`: long name
- `char *description`: full description

The structure `pm_info2_t` contains the following members:

- `int maxpmcs`: number of available counters
- `int *maxevents`: number of events for each hw counter
- `pm_events_t **list_events`: list of available events
- `int thresholdmult`: threshold multiplier
- `int hthresholdmult`: upper threshold multiplier
- `int Hthresholdmult`: hyper threshold multiplier
- `char *proc_name`: processor name
- `pm_events2_t **list_events`: list of available events
- `int thresholdmult`: threshold multiplier
- `int hthresholdmult`: upper threshold multiplier
- `int Hthresholdmult`: hyper threshold multiplier
- `char *proc_name`: processor name
- `pm_feature_t proc_feature`: processor feature list

The structure `pm_events2_t` contains the following members:

- `int event_id`: event number
- `pm_status_t status`: event status
- `char *short_name`: mnemonic name
- `char *long_name`: long name
- `char *description`: full description

The structure `pm_status_t` contains the following members:

- `unsigned:1 b.unverified`: unverified event
- `unsigned:1 b.verified`: verified event
- `unsigned:1 b.caveat`: event verified with some caveat (see description)
- `unsigned:1 b.group_only`: event can only be used within a group
- `unsigned:1 b.threshold`: event can be used with a threshold
- `unsigned:1 b.shared`: event is shared between hardware threads
- `unsigned:1 b.support_mode`: support user, kernel, hypervisor and pmtree counting modes applied
- `unsigned:1 b.overflow`: generate an interrupt on overflow
- `unsigned:1 b.marked`: marked event

The structure `pm_feature_t` contains the following members:

- `unsigned:1 b.hypervisor`: hypervisor counting mode is supported

The structure `pm_groups_info_t` contains the following members:

- `int maxgroups`: number of available groups
- `pm_groups_t *event_groups`: list of event groups

The structure `pm_groups_t` contains the following members:

- `int group_id`: group number
- `int *events`: events in this group, by ID #
- `char *short_name`: mnemonic name
- `char *long_name`: long name
- `char *description`: full description
The structure `pm_prog_t` contains the following members:

- `unsigned:6 mode.b.threshold`: threshold value
- `unsigned:1 mode.b.thresh_res`: uses upper threshold multiplier if set
- `unsigned:1 mode.b.thresh_hres`: uses hyper threshold multiplier if set
- `unsigned:1 mode.b.runlatch`: runlatch enable/disable
- `unsigned:1 mode.b.is_group`: is an event group
- `unsigned:1 mode.b.process`: process level group indicator
- `unsigned:1 mode.b.hypervisor`: turns hypervisor mode counting on
- `unsigned:1 mode.b.kernel`: turns kernel mode counting on
- `unsigned:1 mode.b.user`: turns user mode counting on
- `unsigned:1 mode.b.count`: counting state
- `unsigned:1 mode.b.runlatch`: runlatch enable/disable
- `unsigned:1 mode.b.is_group`: is an event group
- `unsigned:1 mode.b.process`: process level group indicator
- `unsigned:1 mode.b.hypervisor`: turns hypervisor mode counting on
- `unsigned:1 mode.b.kernel`: turns kernel mode counting on
- `unsigned:1 mode.b.user`: turns user mode counting on
- `unsigned:1 mode.b.count`: counting state
- `int events[MAX_COUNTERS]`: list of counted events

The structure `pm_data_t` contains the following members:

- `pm_ginfo_t ginfo`: group information
- `long long accu[MAX_COUNTERS]`: accumulated data

The structure `pm_ginfo_t` contains the following members:

- `int members;`: number of threads in group
- `unsigned:1 flags.b.process`: process level group indicator
- `unsigned:1 flags.b.consistent`: group data consistent with members

The structure `pm_accu_time_t` contains the following members:

- `timebasestruct_t accu_timebase`: accumulated time
- `timebasestruct_t accu_purr`: accumulated PURR time
- `timebasestruct_t accu_spurr`: accumulated SPURR time

Type for timeslice:

```c
typedef int pm_events_prog_t[MAX_COUNTERS] array of counted events
```

The structure `pm_prog_mx_t` contains the following members:

- `unsigned:6 mode.b.threshold`: threshold value
- `unsigned:1 mode.b.thresh_res`: uses upper threshold multiplier if set
- `unsigned:1 mode.b.thresh_hres`: uses hyper threshold multiplier if set
- `unsigned:1 mode.b.runlatch`: runlatch enable/disable
- `unsigned:1 mode.b.is_group`: is an event group
- `unsigned:1 mode.b.process`: process level group indicator
- `unsigned:1 mode.b.hypervisor`: turns hypervisor mode counting on
- `unsigned:1 mode.b.kernel`: turns kernel mode counting on
- `unsigned:1 mode.b.user`: turns user mode counting on
- `unsigned:1 mode.b.count`: counting state
- `int slice_duration`: duration of each time slice in ms
- `int nb_events_prog`: number of counted event sets
- `pm_events_prog_t *events_set;`: array of counted event sets

The structure `pm_prog_mm_t` contains the following members:

- `int proctree`: turns process tree counting on
- `int slice_duration`: duration of each time slice in ms
- `int nb_set_prog`: number of counted event sets
- `pm_prog_t *prog_set;`: array of counted event sets / counting mode

The structure `pm_accu_mx_t` contains the following members:

- `timebasestruct_t accu_time`: accumulated time
- `timebasestruct_t accu_purr`: accumulated PURR time
- `timebasestruct_t accu_spurr`: accumulated SPURR time
- `long long accu_data[MAX_COUNTERS]`: accumulated data

The structure `pm_data_mx_t` contains the following members:
pm_ginfo_t ginfo  group information
int nb_accu_mx  number of data accumulator sets
int nb_mx_round  number of loops on all the event sets
pm_accu_mx_t *accu_set  array of data accumulator sets

The following are declared as functions:

double pm_cycles(void)
void pm_error(char *where, int error)
int pm_get_proctype()

int pm_init(int filter, pm_info_t *pminfo, pm_groups_info_t *pmgroupsinfo)
int pm_initialize(int filter, pm_info2_t *pminfo, pm_groups_info_t *pmgroups, int proctype)
int pm_set_program(pm_prog_t *prog)
int pm_get_program(pm_prog_t *prog)
int pm_start(void)
int pm_tstart(timebasestruct_t *time)
int pm_stop(void)
int pm_tstop(timebasestruct_t *time)
int pm_reset_data(void)
int pm_get_data(pm_data_t *data)
int pm_get_tdata(pm_data_t *data, timebasestruct_t *time)
int pm_get_Tdata(pm_data_t *data, pm_accu_time_t *time)
int pm_get_data_cpu(int cpuid, pm_data_t *data)
int pm_get_tdata_cpu(int cpuid, pm_data_t *data, timebasestruct_t *time)
int pm_get_Tdata_cpu(int cpuid, pm_data_t *data, pm_accu_time_t *time)
int pm_get_data_cpu(int cpuid, pm_data_t *data)
int pm_get_tdata_cpu(int cpuid, pm_data_t *data, timebasestruct_t *time)
int pm_get_Tdata_cpu(int cpuid, pm_data_t *data, pm_accu_time_t *time)
int pm_delete_program(void)
int pm_set_program_mythread(pm_prog_t *prog)
int pm_get_program_mythread(pm_prog_t *prog)
int pm_start_mythread(void)
int pm_tstart_mythread(timebasestruct_t *time)
int pm_stop_mythread(void)
int pm_tstop_mythread(timebasestruct_t *time)
int pm_reset_data_mythread(void)
int pm_get_data_mythread(pm_data_t *data)
int pm_get_tdata_mythread(pm_data_t *data, timebasestruct_t *time)
int pm_get_Tdata_mythread(pm_data_t *data, pm_accu_time_t *time)
int pm_delete_program_mythread(void)
int pm_get_program_mygroup(pm_prog_t *prog)
int pm_set_program_mygroup(pm_prog_t *prog)
int pm_start_mygroup(void)
int pm_tstart_mygroup(timebasestruct_t *time)
int pm_stop_mygroup(void)
int pm_tstop_mygroup(timebasestruct_t *time)
int pm_reset_data_mygroup(void)
int pm_get_data_mygroup(pm_data_t *data)
int pm_get_tdata_mygroup(pm_data_t *data, timebasestruct_t *time)
int pm_get_Tdata_mygroup(pm_data_t *data, pm_accu_time_t *time)
int pm_delete_program_mygroup(void)

int pm_set_program_thread(pid_t pid, tid_t tid, pm_prog_t *prog)
int pm_get_program_thread(pid_t pid, tid_t tid, pm_prog_t *prog)
int pm_start_thread(pid_t pid, tid_t tid)
int pm_tstart_thread(pid_t pid, tid_t tid, timebasestruct_t *time)
int pm_stop_thread(pid_t pid, tid_t tid)
int pm_tstop_thread(pid_t pid, tid_t tid, timebasestruct_t *time)
int pm_reset_data_thread(pid_t pid, tid_t tid)
int pm_get_data_thread(pid_t pid, tid_t tid, pm_data_t *data)
int pm_get_tdata_thread(pid_t pid, tid_t tid, pm_data_t *data, timebasestruct_t *time)
int pm_get_Tdata_thread(pid_t pid, tid_t tid, pm_data_t *data, pm_accu_time_t *time)
int pm_delete_program_thread(pid_t pid, tid_t tid)

int pm_set_program_mygroup(pm_prog_t *prog)
int pm_get_program_mygroup(pm_prog_t *prog)
int pm_start_mygroup(void)
int pm_tstart_mygroup(timebasestruct_t *time)
int pm_stop_mygroup(void)
int pm_tstop_mygroup(timebasestruct_t *time)
int pm_reset_data_mygroup(void)
int pm_get_data_mygroup(pm_data_t *data)
int pm_get_tdata_mygroup(pm_data_t *data, timebasestruct_t *time)
int pm_get_Tdata_mygroup(pm_data_t *data, pm_accu_time_t *time)
int pm_delete_program_mygroup(void)
Chapter 4. Header Files

int pm_set_program_group(pid_t pid, tid_t tid, pm_prog_t *prog)
int pm_get_program_group(pid_t pid, tid_t tid, pm_prog_t *prog)
int pm_start_group(pid_t pid, tid_t tid)
int pm_tstart_group(pid_t pid, tid_t tid, timebasestruct_t *time)
int pm_stop_group(pid_t pid, tid_t tid)
int pm_tstop_group(pid_t pid, tid_t tid, timebasestruct_t *time)
int pm_reset_data_group(pid_t pid, tid_t tid, pm_data_t *data)
int pm_get_data_group(pid_t tid, pm_data_t *data, pm_accu_time_t *time)
int pm_delete_program_group(pid_t pid, tid_t tid)

int pm_set_program_pgroup(pid_t pid, tid_t tid, ptid_t ptid, pm_prog_t *prog)
int pm_get_program_pgroup(pid_t pid, tid_t tid, ptid_t ptid, pm_prog_t *prog)
int pm_start_pgroup(pid_t pid, tid_t tid, ptid_t ptid)
int pm_tstart_pgroup(pid_t pid, tid_t tid, ptid_t ptid, timebasestruct_t *time)
int pm_stop_pgroup(pid_t pid, tid_t tid, ptid_t ptid)
int pm_tstop_pgroup(pid_t pid, tid_t tid, ptid_t ptid, timebasestruct_t *time)
int pm_reset_data_pgroup(pid_t pid, tid_t tid, ptid_t ptid, pm_data_t *data)
int pm_get_data_pgroup(pid_t tid, pm_data_t *data, pm_accu_time_t *time)
int pm_delete_program_pgroup(pid_t pid, tid_t tid, ptid_t ptid)

int pm_set_program_mx(pm_prog_mx_t *prog)
int pm_get_program_mx(pm_prog_mx_t *prog)
int pm_set_program_mm(pm_prog_mm_t *prog)
int pm_get_program_mm(pm_prog_mm_t *prog)
int pm_set_program_mythread_mm(pm_prog_mm_t *prog)
int pm_get_program_mythread_mm(pm_prog_mm_t *prog)
int pm_set_program_mythread_mx(pm_data_mx_t *data)
int pm_get_tdata_mythread_mx(pm_data_mx_t *data, pm_accu_time_t *time)

int pm_set_program_thread_mx(pid_t pid, tid_t tid, pm_prog_mx_t *prog)
int pm_get_program_thread_mx(pid_t pid, tid_t tid, pm_prog_mx_t *prog)
int pm_set_program_thread_mm(pid_t pid, tid_t tid, pm_prog_mm_t *prog)
int pm_get_program_thread_mm(pid_t pid, tid_t tid, pm_prog_mm_t *prog)
int pm_set_program_thread_mythread_mm(pm_prog_mm_t *prog)
int pm_get_program_thread_mythread_mm(pm_prog_mm_t *prog)
int pm_set_program_thread_mythread_mx(pm_data_mx_t *data)
int pm_get_tdata_thread_mx(pid_t pid, tid_t tid, pm_data_mx_t *data, pm_accu_time_t *time)

int pm_set_program_pgroup_mx(pm_prog_mx_t *prog)
int pm_get_program_mygroup_mx(pm_prog_mx_t *prog)
int pm_set_program_mygroup_mm(pm_prog_mm_t *prog)
int pm_get_data_mygroup_mx(pm_data_mx_t *data)
int pm_get_tdata_mygroup_mx(pm_data_mx_t *data, timebasestruct_t *time)

int pm_set_program_pgroup_mx(pid_t pid, tid_t tid, pm_prog_mx_t *prog)
int pm_get_program_pgroup_mx(pid_t pid, tid_t tid, pm_prog_mx_t *prog)
int pm_set_program_pgroup_mm(pid_t pid, tid_t tid, pm_prog_mm_t *prog)
int pm_get_program_pgroup_mm(pid_t pid, tid_t tid, pm_prog_mm_t *prog)
int pm_get_program_pgroup_mm(pid_t pid, tid_t tid, pm_prog_mm_t *prog)
int pm_set_program_pgroup_mm(pid_t pid, tid_t tid, pm_prog_mm_t *prog)
int pm_get_data_pgroup_mx(pid_t pid, tid_t tid, pm_data_mx_t *data)
int pm_get_tdata_pgroup_mx(pid_t pid, tid_t tid, pm_data_mx_t *data, timebasestruct_t *time)

int pm_set_program_pthread_mx(pid_t pid, tid_t tid, ptid_t ptid, pm_prog_mx_t *prog)
int pm_get_program_pthread_mx(pid_t pid, tid_t tid, ptid_t ptid, pm_prog_mx_t *prog)
int pm_set_program_pthread_mx(pid_t pid, tid_t tid, ptid_t ptid, pm_prog_mx_t *prog)
int pm_get_program_pthread_mx(pid_t pid, tid_t tid, ptid_t ptid, pm_prog_mx_t *prog)
int pm_set_program_pthread_mx(pid_t pid, tid_t tid, ptid_t ptid, pm_prog_mx_t *prog)
int pm_get_data_pthread_mx(pid_t pid, tid_t tid, ptid_t ptid, pm_data_mx_t *data)
int pm_get_tdata_pthread_mx(pid_t pid, tid_t tid, ptid_t ptid, pm_data_mx_t *data, timebasestruct_t *time)

int pm_get_tdata_pgroup_mx(pid_t pid, tid_t tid, ptid_t ptid, pm_data_mx_t *data)
int pm_get_tdata_pgroup_mx(pid_t pid, tid_t tid, ptid_t ptid, pm_data_mx_t *data, timebasestruct_t *time)

Related Information
The pm_cycles subroutine, pm_error subroutine, pm_init subroutine.

The pm_set_program subroutine, pm_get_program subroutine, pm_delete_program subroutine,
pm_get_data subroutine, pm_get_tdata subroutine, pm_get_data_cpu subroutine, pm_get_tdata_cpu subroutine,
pm_get_data_lcpu subroutine, pm_get_tdata_lcpu subroutine, pm_get_Tdata_lcpu subroutine,
pm_start subroutine, pm_stop subroutine, pm_reset_data subroutine.

The pm_set_program_mythread subroutine, pm_get_program_mythread subroutine,
pm_delete_program_mythread subroutine, pm_get_data_mythread subroutine,
pm_get_tdata_mythread subroutine, pm_start_mythread subroutine, pm_stop_mythread subroutine,
pm_reset_data_mythread subroutine.

The pm_set_program_mygroup subroutine, pm_get_program_mygroup subroutine,
pm_delete_program_mygroup subroutine, pm_get_data_mygroup subroutine,
pm_get_tdata_mygroup subroutine, pm_start_mygroup subroutine, pm_stop_mygroup subroutine,
pm_reset_data_mygroup subroutine.

The pm_set_program_thread subroutine, pm_get_program_thread subroutine,
pm_delete_program_thread subroutine, pm_get_data_thread subroutine,
pm_get_tdata_thread subroutine, pm_start_thread subroutine, pm_stop_thread subroutine,
pm_reset_data_thread subroutine.

The pm_set_program_pgroup subroutine, pm_get_program_pgroup subroutine,
pm_delete_program_pgroup subroutine, pm_get_data_pgroup subroutine,
pm_get_tdata_pgroup subroutine, pm_start_pgroup subroutine, pm_stop_pgroup subroutine,
pm_reset_data_pgroup subroutine.
pm_reset_data_pthread subroutine, pm_reset_data_pgroup subroutine, pm_get_data_pthread subroutine, pm_get_data_pgroup subroutine, pm_get_tdata_pthread subroutine, pm_get_tdata_pgroup subroutine, pm_delete_program_pthread subroutine, and the pm_delete_program_pgroup subroutine.

The pm_set_program_mx subroutine, pm_get_program_mx subroutine, pm_set_program_mm subroutine, pm_get_program_mm subroutine, pm_get_data_mx subroutine, pm_get_tdata_mx subroutine, pm_delete_program_pthread subroutine, and the pm_delete_program_pgroup subroutine.

The pm_set_program_mythread_mx subroutine, pm_get_program_mythread_mx subroutine, pm_set_program_mythread_mm subroutine, pm_get_program_mythread_mm subroutine, pm_get_data_mythread_mx subroutine, pm_get_tdata_mythread_mx subroutine, pm_delete_program_pthread subroutine, and the pm_delete_program_pgroup subroutine.

The pm_set_program_mygroup_mx subroutine, pm_get_program_mygroup_mx subroutine, pm_set_program_mygroup_mm subroutine, pm_get_program_mygroup_mm subroutine, pm_get_data_mygroup_mx subroutine, pm_get_tdata_mygroup_mx subroutine, pm_delete_program_pthread subroutine, and the pm_delete_program_pgroup subroutine.

The pm_set_program_thread_mx subroutine, pm_get_program_thread_mx subroutine, pm_set_program_thread_mm subroutine, pm_get_program_thread_mm subroutine, pm_get_data_thread_mx subroutine, pm_get_tdata_thread_mx subroutine, pm_delete_program_pthread subroutine, and the pm_delete_program_pgroup subroutine.

The pm_set_program_pthread_mx subroutine, pm_set_program_pgroup_mx subroutine, pm_get_program_pthread_mx subroutine, pm_get_program_pgroup_mx subroutine, pm_set_program_pthread_mm subroutine, pm_set_program_pgroup_mm subroutine, pm_get_program_pthread_mm subroutine, pm_get_program_pgroup_mm subroutine, pm_get_data_pthread_mx subroutine, pm_get_tdata_pthread_mx subroutine, pm_get_data_pgroup_mx subroutine, pm_get_tdata_pgroup_mx subroutine.

poll.h File

Purpose
Defines the structures and flags used by the poll subroutine.

Description
The /usr/include/sys/poll.h file defines several structures used by the poll subroutine. An array of pollfd or pollmsg structures or a pollist structure specify the file descriptors or pointers and message queues for which the poll subroutine checks the I/O status. This file also defines the returned events flags, error events flags, device-type flags and input flags used in polling operations.

During a polling operation on both file descriptors and message queues, the ListPointer parameter points to a pollist structure, which can specify either file descriptors or pointers and message queues. The program must define the pollist structure in the following form:

struct pollist {
    struct pollfd fdlist[f];
    struct pollmsg msglist[m];
};

The pollfd structure and the pollmsg structure in the pollist structure perform the following functions:

pollfd[f] This structure defines an array of file descriptors or file pointers. The f variable specifies the number of elements in the array.
This structure defines an array of message queue identifiers. The \( m \) variable specifies the number of elements in the array.

A \texttt{POLLIST} macro is also defined in the \texttt{poll.h} file to define the \texttt{pollist} structure. The format of the macro is:

\texttt{POLLIST(f, m) Declarator . . . ;}

The \textit{Declarator} parameter is the name of the variable that is declared as having this type.

The \texttt{pollfd} and \texttt{pollmsg} structures defined in the \texttt{poll.h} file contain the following fields:

- \texttt{fd} Specifies a valid file descriptor or file pointer to the \texttt{poll} subroutine. If the value of this field is negative, this element is skipped.
- \texttt{msgid} Specifies a valid message queue ID to the \texttt{poll} subroutine. If the value of this field is negative, this element is skipped.
- \texttt{events} The events being tracked. This is any combination of the following flags:
  - \texttt{POLLIN} Input is present on the file or message queue.
  - \texttt{POLLOUT} The file or message queue is capable of accepting output.
  - \texttt{POLLPRI} An exceptional condition is present on the file or message queue.
- \texttt{revents} Returned events. This field specifies the events that have occurred. This can be any combination of the events requested by the events field. This field can also contain one of the following flags:
  - \texttt{POLLNVAL} The value specified by the \texttt{fd} field or the \texttt{msgid} field is neither a valid file descriptor or pointer nor the identifier of an accessible message queue.
  - \texttt{POLLERR} An error condition arose on the specified file or message queue.

\textbf{Related Information}

The \texttt{fp_poll} kernel service, \texttt{fp_select} kernel service, \texttt{selnotify} kernel service.

The \texttt{poll} subroutine, \texttt{select} subroutine.

The \texttt{Input and Output Handling Programmer's Overview} in \textit{AIX 5L Version 5.3 General Programming Concepts: Writing and Debugging Programs} describes the files, commands, and subroutines used for low-level, stream, terminal, and asynchronous I/O interfaces.

\textbf{pthread.h File}

\textbf{Purpose}

Lists threads.

\textbf{Syntax}

\texttt{#include <pthread.h>}

\textbf{Description}

The \texttt{pthread.h} header defines the following symbols:
The pthread_attr_t, pthread_cond_t, pthread_condattr_t, pthread_key_t, pthread_mutex_t, pthread_mutexattr_t, pthread_once_t, pthread_rwlock_t, pthread_rwlockattr_t, and pthread_t types are defined as described in sys/types.h.

The following are declared as functions and may also be declared as macros. Function prototypes must be provided for use with an ISO C compiler.

```c
int pthread_attr_destroy (pthread_attr_t *);
int pthread_attr_getdetachstate (const pthread_attr_t *, int *);
int pthread_attr_getguardsize (const pthread_attr_t *, size_t *);
int pthread_attr_getinheritsched (const pthread_attr_t *, int *);
int pthread_attr_getschedparam (const pthread_attr_t *, struct sched_param *);
int pthread_attr_getschedpolicy (const pthread_attr_t *, int *);
int pthread_attr_getscope (const pthread_attr_t *, int *);
int pthread_attr_getstackaddr (const pthread_attr_t *, void **);
int pthread_attr_getstacksize (const pthread_attr_t *, size_t *);
int pthread_attr_init (pthread_attr_t *);
int pthread_attr_setdetachstate (pthread_attr_t *, int);
int pthread_attr_setguardsize (pthread_attr_t *, size_t);
int pthread_attr_setinheritsched (pthread_attr_t *, int);
int pthread_attr_setschedparam (pthread_attr_t *, struct sched_param *);
int pthread_attr_setschedpolicy (pthread_attr_t *, int);
int pthread_attr_setscope (pthread_attr_t *, int);
int pthread_attr_setstackaddr (pthread_attr_t *, void *);
int pthread_attr_setstacksize (pthread_attr_t *, size_t);
int pthread_cancel (pthread_t);
void pthread_cleanup_push (void (*)(void*), void *);
void pthread_cleanup_pop (int);
int pthread_cond_broadcast (pthread_cond_t *);
int pthread_cond_destroy (pthread_cond_t *);
int pthread_cond_init (pthread_cond_t *, const pthread_condattr_t *);
int pthread_cond_signal (pthread_cond_t *);
int pthread_cond_timedwait (pthread_cond_t *, pthread_mutex_t *, const struct timespec *);
int pthread_cond_wait (pthread_cond_t *);
int pthread_condattr_destroy (pthread_condattr_t *);
int pthread_condattr_getpshared (const pthread_condattr_t *, int *);
int pthread_condattr_init (pthread_condattr_t *);
int pthread_condattr_setpshared (pthread_condattr_t *, int);
int pthread_create (pthread_t *, const pthread_attr_t *, void *(*)(void*), void *);
int pthread_detach (pthread_t);
int pthread_equal (pthread_t, pthread_t);
```
Inclusion of the pthread.h header will make visible symbols defined in the headers sched.h and time.h.

Related Information

The pthread_attr_init, pthread_attr_getguardsize, pthread_attr_setscope, pthread_cancel,
pthread_cleanup_push, pthread_cond_init, pthread_cond_signal, pthread_cond_wait,
pthread_condattr_init, pthread_create, pthread_detach, pthread_equal, pthread_exit,
pthread_getconcurrency, pthread_getschedparam, pthread_join, pthread_key_create,
pthread_key_delete, pthread_mutex_init, pthread_mutex_lock, pthread_mutex_setprioceiling,
pthread_mutexattr_init, pthread_mutexattr_getprioceiling, pthread_mutexattr_setprioceiling,
pthread_once, pthread_rwlock_destroy, pthread_rwlock_getprioceiling, pthread_rwlock_setprioceiling,
subroutines.
pwd.h File

Purpose
Describes password structure.

Syntax
#include <pwd.h>

Description
The `pwd.h` header provides a definition for `struct passwd`, which includes at least the following members:

- `char *pw_name` user's login name
- `uid_t pw_uid` numerical user ID
- `gid_t pw_gid` numerical group ID
- `char *pw_dir` initial working directory
- `char *pw_shell` program to use as shell

The `gid_t` and `uid_t` types are defined as described in `sys/types.h`.

The following are declared as functions and may also be defined as macros. Function prototypes must be provided for use with an ISO C compiler.

```c
struct passwd *getpwuid(uid_t);
int getpwnam_r(const char *, struct passwd *, char *, size_t, struct passwd **);
int getpwnam_r(uid_t, struct passwd *, char *, size_t, struct passwd **);
void endpwent(void);
struct passwd *getpwent(void);
void setpwent(void);
```

Related Information
The `endpwent`, `getpwnam`, `getpwuid` and `getpwuid_r` subroutines.

The `sys/types.h` header file.

pwdpolicy.h File

Purpose
Defines the types and manifest constants required to support the `passwdpolicy()` function.

Description
The format of the `pwdpolicy.h` header file shall be similar to the password construction rule attributes as stored in the `/etc/security/user` file, with the exception that named policies do not include the histsize and histexpire attributes. Each file is a sequence of zero or more stanzas with the named policy being the stanza name. Each stanza contains one or more attributes describing the password rules which must be satisfied for a password to be accepted.

Password policy parameters are in the following table.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>pwp_version</code></td>
<td>Specifies the version of the <code>passwd_policy_t</code> structure. The current structure version number is <code>PWP_VERSION_1</code>. Future extensions to this structure will use a different version number.</td>
</tr>
<tr>
<td><code>pwp_minage</code></td>
<td>The number of seconds as a time32_t between the time a password is modified and the time the password may again be modified. This field is referenced if <code>PWP_TOO_SOON</code> is set in checks.</td>
</tr>
</tbody>
</table>
### pwp_maxage
The number of seconds as a `time32_t` after a password has been modified when it is considered to be expired. This field is referenced if `PWP_EXPIRED` is set in checks.

### pwp_maxexpired
The number of seconds, as a `time32_t`, after a password has expired when it may no longer be modified. A value of 0 indicates that an expired password may not be changed. A value of -1 indicates that an expired password may be changed after any length of time. This field is referenced if `PWP_EXPIRED` is set in checks.

### pwp_minalpha
The minimum number of characters in the password which must be alphabetic characters, as determined by invoking the `isalpha()` macro. A value less than or equal to 0 disables this test. This field is referenced if `PWP_TOO_FEW_ALPHA` is set in checks.

### pwp_minother
The minimum number of characters in the password which may not be alphabetic characters, as determined by invoking the `isalpha()` macro. A value less than or equal to 0 disables this test. This field is referenced if `PWP_TOO_FEW_OTHER` is set in checks.

### pwp_minlen
The minimum total number of characters in the password. A value less than or equal to 0 disables this test. This field is referenced if `PWP_TOO_SHORT` is set in checks.

### pwp_maxrepeats
The maximum number of times an individual character may appear in the password. A value less than or equal to 0 disables this test. This field is referenced if `PWP_TOO_MANY_REPEATS` is set in checks.

### pwp_mindiff
The minimum number of characters which must be changed between the old password and the new password. A value less than or equal to 0 disables this test. This field is referenced if `PWP_TOO_MANY_SAME` is set in checks.

## Example

```c
#include <sys/types.h>
/
/* Name types */
#define PWP_USERNAME 1
#define PWP_SYSTEMPOLICY 2
#define PWP_LOCALPOLICY 3
/* Test flag values */
#define PWP_TOO_SOON 0x0001
#define PWP_EXPIRED 0x0002
#define PWP_TOO_FEW_ALPHA 0x0004
#define PWP_TOO_FEW_OTHER 0x0008
#define PWP_TOO_SHORT 0x0010
#define PWP_TOO_MANY_REPEATS 0x0020
#define PWP_TOO_MANYSAME 0x0040
#define PWP_IN_DICTIONARY 0x0080
#define PWP_REUSED_PW 0x0100
#define PWP_REUSED_TOO_SOON 0x0200
#define PWP_FAILED_OTHER 0x0400
/* Policy structure version number */
#define PWP_VERSION_1 1
/* Policy structure definition */
typedef struct {
    int pwp_version;
    time32_t pwp_minage;
    time32_t pwp_maxage;
    time32_t pwp_maxexpired;
    int pwp_minalpha;
    int pwp_minother;
    int pwp_minlen;
    int pwp_maxrepeats;
    int pwp_mindiff;
} passwd_policy_t;
```

The `maxage`, `minage`, `maxexpired`, `maxrepeats`, `mindiff`, `minalpha`, `minother`, and `minlen` attributes are integers. The `dictionlist` and `pwdchecks` attributes are comma-separated lists of filenames. For more information on valid values for attributes, please see `/etc/security/user`.
Permissions
Only the root user should have write (w) access.

Location
/usr/include/pwdpolicy.h

Related Information
The `/etc/security/user` file.
The `/usr/lib/security/passwd_policy` file.

sem.h File

Purpose
Describes the structures that are used by subroutines that perform semaphore operations.

Description
The `/usr/include/sys/semp.h` file defines the structures that are used by the `semop` subroutine and the `semctl` subroutine to perform various semaphore operations.

The `sem` structure stores the values that the `Commands` parameter of the `semctl` subroutine gets and sets. This structure contains the following fields:

- `semval` Specifies the operation permission structure of a semaphore. The data type of this field is unsigned short.
- `sempid` Specifies the last process that performed a `semop` subroutine. The data type of this field is `pid_t`.
- `semncnt` Specifies the number of processes awaiting `semval > cval`. The data type of this field is unsigned short.
- `semzcnt` Specifies the number of processes awaiting `semval = 0`. The data type of this field is unsigned short.

The `sembuf` structure stores semaphore information used by the `semop` subroutine. This structure contains the following fields:

- `sem_num` Specifies a semaphore on which to perform some semaphore operation. The data type of this field is unsigned short.
- `sem_op` Specifies a semaphore operation to be performed on the semaphore specified by the `sem_num` field and the `SemaphoreID` parameter of the `semop` subroutine. This value can be a positive integer, a negative integer, or 0:

  - `i` If the current process has write permission, the positive integer value of this field is added to the value of the `semval` field of the semaphore.
  - `-i` If the current process has write permission, a negative integer value in this field causes one of the following actions:

    - If the `semval` field is greater than or equal to the absolute value of the `sem_op` field, the absolute value of the `sem_op` field is subtracted from the value of the `semval` field.
    - If the `semval` field is less than the absolute value of the `sem_op` field and the `IPC_NOWAIT` flag is set, the `semop` subroutine returns a value of -1 and sets the `errno` global variable to `EAGAIN`. 
If the value of the `semval` field is less than the absolute value of the `sem_op` field and the `IPC_NOWAIT` flag is not set, the `semop` subroutine increments the `semncnt` field associated with the specified semaphore and suspends execution of the calling process until one of the following conditions is met:

- The value of the `semval` field becomes greater than or equal to the absolute value of the `sem_op` field. When this occurs, the value of the `semncnt` field associated with the specified semaphore is decremented, the absolute value of the `sem_op` field is subtracted from `semval` value and, if the `SEM_UNDO` flag is set in the `sem_flg` field, the absolute value of the `sem_op` field is added to the `Semadj` value of the calling process for the specified semaphore.
- The semaphore specified by the `SemaphoreID` parameter for which the calling process is awaiting action is removed from the system (see the `semctl` subroutine). When this occurs, the `errno` global variable is set equal to `EIDRM`, and a value of -1 is returned.
- The calling process receives a signal that is to be caught. When this occurs, the value of the `semncnt` field associated with the specified semaphore is decremented, and the calling process resumes execution in the manner prescribed in the `sigaction` subroutine.

0  If the current process has read permission, a value of 0 in this field causes one of the following actions:

- If the `semval` field is 0, the `semop` subroutine returns a value of 0.
- If the `semval` field is not equal to 0 and the `IPC_NOWAIT` flag is set, the `semop` subroutine returns a value of -1 and sets the `errno` global variable to `EAGAIN`.
- If `semval` is not equal to 0 and the `IPC_NOWAIT` flag is not set, the `semop` subroutine increments the `semzcnt` field associated with the specified semaphore and suspends execution of the calling process until one of the following conditions is met:
  - The value of the `semval` field becomes 0, at which time the value of the `semzcnt` field associated with the specified semaphore is decremented.
  - The semaphore specified by the `SemaphoreID` parameter for which the calling process is awaiting action is removed from the system. When this occurs, the `errno` global variable is set equal to `EIDRM`, and a value of -1 is returned.
  - The calling process receives a signal that is to be caught. When this occurs, the value of the `semzcnt` field associated with the specified semaphore is decremented, and the calling process resumes execution in the manner prescribed in the `sigaction` subroutine.

The data type of the `sem_op` field is short.

`sem_flg`

If the value of this field is not 0 for an operation, the value is constructed by logically ORing one or more of the following values:

**SEM_UNDO**

Specifies whether to modify the `Semadj` values of the calling process.

If this value is set for an operation and the value of the `sem_op` field is a positive integer, the value of the `sem_op` field is subtracted from the `Semadj` value of the calling process.

If this value is set for an operation and the value of the `sem_op` field is a negative integer, the absolute value of the `sem_op` field is added to the `Semadj` value of the calling process. The `exit` subroutine adds the `Semadj` value to the value of the `semval` field of the semaphore when the process terminates.

**SEM_ORDER**

Specifies whether to perform atomically or individually the operations specified by the
SemaphoreOperations array of the semop subroutine. (This flag is valid only when included in the SemaphoreOperations[0].sem_flg parameter, the first operation in the SemaphoreOperations array.)

If the SEM_ORDER flag is not set (the default), the specified operations are performed atomically. That is, none of the semval values in the array are modified until all of the semaphore operations are completed. If the calling process must wait until some semval requirement is met, the semop subroutine does so before performing any of the operations. If any semaphore operation would cause an error to occur, none of the operations are performed.

If the SEM_ORDER flag is set, the operations are performed individually in the order that they appear in the array, regardless of whether any of the operations require the process to wait. If an operation encounters an error condition, the semop subroutine sets the SEM_ERR flag in the sem_flg field of the failing operation; neither the failing operation nor the following operations in the array are performed.

**IPC_NOWAIT**

Specifies whether to wait or to return immediately when the semval of a semaphore is not a certain value.

The data type of the sem_flg field is short.

The semid_ds structure stores semaphore status information used by the semctl subroutine and pointed to by the Buffer parameter. This structure contains the following fields:

- **sem_perm**: Specifies the operation permission structure of a semaphore. The data type of this field is struct ipc_perm.
- **sem_nsems**: Specifies the number of semaphores in the set. The data type of this field is unsigned short.
- **sem_otime**: Specifies the time at which a semop subroutine was last performed. The data type of this field is time_t.
- **sem_ctime**: Specifies the time at which this structure was last changed with a semctl subroutine. The data type of this field is time_t.

Related Information

The atexit subroutine, exec subroutines, exit subroutine, fork subroutine, semct1 subroutine, semget subroutine, semop subroutine, sigaction subroutine.

---

**sgtty.h File**

**Purpose**

Provides the terminal interface for the Berkeley line discipline.

**Description**

The sgtty.h file defines the structures used by ioctl subroutines that apply to terminal files. The structures, definitions, and values in this file are provided for compatibility with the Berkeley user interface for asynchronous communication. Window and terminal size operations use the winsize structure, which is defined in the ioctl.h file. The winsize structure and the ioctl functions that use it are described in [tty Subsystem Overview] in AIX 5L Version 5.3 General Programming Concepts: Writing and Debugging Programs.

**Note**: Version 4 supports the Berkeley line discipline for compatibility with older applications. However, it is strongly recommended to use the POSIX compliant line discipline, which interface is described in the termios.h file.
**Basic sgty.h Modes**

Basic ioctl functions use the `sgttyb` structure defined in the `sgtty.h` file. This structure contains the following fields:

**sg_ispeed**

Specifies the input speed of the device. For any particular hardware, impossible speed changes are ignored. Symbolic values in the table are as defined in the `sgtty.h` file.

- **B0**  Hangs up. The zero baud rate is used to hang up the connection. If B0 is specified, the `data terminal ready` signal is dropped. As a result, the line is usually disconnected.
- **B50**  50 baud.
- **B75**  75 baud.
- **B110** 110 baud.
- **B134** 134.5 baud.
- **B150** 150 baud.
- **B200** 200 baud.
- **B300** 300 baud.
- **B600** 600 baud.
- **B1200** 1200 baud.
- **B1800** 1800 baud.
- **B2400** 2400 baud.
- **B4800** 4800 baud.
- **B9600** 9600 baud.
- **EXTA**  External A.
- **EXTB**  External B.

**sg_ospeed**

Specifies the output speed of the device. Refer to the description of the `sg_ispeed` field. The `sg_ospeed` field has the same values as the `sg_ispeed` field.

**sg_erase**

Specifies the erase character. (The default is Backspace.)

**sg_kill**

Specifies the kill character. (The default is Ctrl-U.)

**sg_flags**

Specifies how the system treats output. The initial output-control value is all bits clear. The possible output modes are:

- **ALLDELAY**  Delays algorithm selection.
- **BSDELAY**  Selects backspace delays. Backspace delays are currently ignored. Possible values are BS0 or BS1.
- **VTDELAY**  Selects form-feed and vertical-tab delays:
  - **FF0**  Specifies no delay.
  - **FF1**  Specifies one delay of approximately 2 seconds.
CRDELAY
Selects carriage-return delays:

- **CR0** Specifies no delay.
- **CR1** Specifies one delay. The delay lasts approximately 0.08 seconds.
- **CR2** Specifies one delay. The delay lasts approximately 0.16 seconds.
- **CR3** Specifies one delay. The delay pads lines to be at least 9 characters at 9600 baud.

TBDELAY
Selects tab delays:

- **TAB0** Specifies no delay.
- **TAB1** Specifies one delay. The delay is dependent on the amount of movement.
- **TAB2** Specifies one delay. The delay lasts about 0.10 seconds.

- **XTABS** Specifies that tabs are to be replaced by the appropriate number of spaces on output.

NLDELAY
Selects the new-line character delays. This is a mask to use before comparing to NL0 and NL1.

- **NL0** Specifies no delay.
- **NL1** Specifies one delay. The delay is dependent on the current column.
- **NL2** Specifies one delay. The delay lasts about 0.10 seconds.

The delay bits specify how long transmission stops to allow for mechanical or other movement when certain characters are sent to the terminal. The actual delays depend on line speed and system load.

EVENP
Allows even parity on input.

The **EVENP** and **ODDP** flags control both parity checking on input and parity generation on output in COOKED and CBREAK mode (unless the LPASS8 bit is enabled). Even parity is generated on output unless the **ODDP** flag is set and the **EVENP** flag is clear, in which case odd parity is generated. Input characters with the wrong parity, as determined by the **EVENP** and **ODDP** flags, are ignored in COOKED and CBREAK mode.

ODDP
Allows odd parity on input. Refer to the description of the **EVENP** flag.

RAW
Indicates the RAW mode, which features a wake up on all characters and an 8-bit interface.

The RAW mode disables all processing except output flushing specified by the **LFLUSH0** bit. The full 8 bits of input are given as soon as they are available; all 8 bits are passed on output. A break condition in the input is reported as a null character. If the input queue overflows in RAW mode, all data in the input and output queues is discarded; this applies to both the new and old drivers.

CRMOD
Maps a carriage return into a new line on input and outputs a new line as a carriage return and a new line.

ECHO
Echo (full duplex).
**LCASE**
Maps uppercase to lowercase on input and lowercase to uppercase on output on uppercase terminals.

**CBREAK**
Enables a half-cooked mode. Programs can read each character as it is typed instead of waiting for a full line. All processing is done except input editing. Character and word erase, line kill, input reprint, and special treatment of the backslash character and the EOT character are disabled.

**TANDEM**
Enables automatic flow control (TANDEM mode), which causes the system to produce a stop character (Ctrl-S) when the input queue is in danger of overflowing, and a start character (Ctrl-Q) when the input queue has drained sufficiently. This mode is useful for flow control when the terminal is actually another computer that understands the conventions.

**Note:** The same stop and start characters are used for both directions of flow control. The character specified by the t_stopc field is accepted on input as the character that stops output and is produced on output as the character to stop input. The character specified by the t_startc field is accepted on input as the character that restarts output and is produced on output as the character to restart input.

**Basic ioctl Operations**
A large number of ioctl commands apply to terminals. Some have the general form:

```c
#include <sgtty.h>
ioctl(FileDescriptor, Code, Value)
struct sgttyb *Value;
```

The applicable values for the **Code** parameter are:

- **TIOCGETP** Fetches the basic parameters associated with the terminal and stores them in the `sgttyb` structure that is pointed to.
- **TIOCSETP** Sets the parameters according to the `sgttyb` structure that is pointed to. The interface delays until output stops, then throws away any unread characters before changing the modes.
- **TIOCSETN** Has the same effect as the **TIOCSETP** value but does not delay or flush input. Input is not preserved, however, when changing to or from the RAW mode.

For the following codes, the **Value** parameter is ignored:

- **TIOCEXCL** Sets exclusive-use mode; no further opens are permitted until the file is closed.
- **TIOCNXCL** Turns off exclusive-use mode.
- **TIOCHPCL** When the file is closed for the last time, hangs up the terminal. This is useful when the line is associated with a modem used to place outgoing calls.

For the following code, the **Value** parameter is a pointer to an integer.

- **TIOCFLUSH** If the integer pointed to by the **Value** parameter has a zero value, all characters waiting in input or output queues are flushed. Otherwise, the value of the integer applies to the FREAD and FWRITE bits defined in the `fcntl.h` file. If the FREAD bit is set, all characters waiting in input queues are flushed. If the FWRITE bit is set, all characters waiting in output queues are flushed.

**Note:** The FREAD and FWRITE bits cannot be used unless the _KERNEL flag is set.
In the following codes, the argument is 0 unless specified otherwise:

- **TIOCSTI**: The `Value` parameter points to a character that the system pretends has been typed on the terminal.
- **TIOCSBRK**: The break bit is set in the terminal.
- **TIOCCBRK**: The break bit is cleared.
- **TIOCSDTR**: Data terminal ready is set.
- **TIOCCDTR**: Data terminal ready is cleared.
- **TIOCSTOP**: Output is stopped as if the stop character had been typed.
- **TIOCSTART**: Output is restarted as if the start character had been typed.
- **TIOCSPGRP**: The `Value` parameter is a pointer to an integer which is the value to which the process group ID for this terminal will be set.
- **TIOCOUTQ**: Returns in the integer pointed to by the `Value` parameter the number of characters queued for output to the terminal.
- **TIONREAD**: Returns in the integer pointed to by the `Value` parameter the number of characters immediately readable from the argument descriptor. This works for files, pipes, and terminals.

**Uppercase Terminals**

If the `LCASE` output-mode bit is set, all uppercase letters are mapped into the corresponding lowercase letter. The uppercase letter can be generated by preceding it with a `\` (backslash). Uppercase letters are preceded by a backslash when they are output. In addition, the following escape sequences can be generated on output and accepted on input:

<table>
<thead>
<tr>
<th>For</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td><code> </code> (grave)</td>
<td><code>\</code></td>
</tr>
<tr>
<td><code>!</code></td>
<td><code>\!</code></td>
</tr>
<tr>
<td><code>~</code></td>
<td><code>\^</code></td>
</tr>
<tr>
<td><code>{</code></td>
<td><code>\</code></td>
</tr>
<tr>
<td><code>}</code></td>
<td><code>\</code></td>
</tr>
</tbody>
</table>

To deal with terminals that do not understand that the `~` (tilde) has been made into an ASCII character, the `LTILDE` bit can be set in the local-mode word. When the `LTILDE` bit is set, the `~` (tilde) character will be replaced with the ` ` (grave) character on output.

**Special Characters**

A `tchars` structure associated with each terminal specifies special characters for both the old and new terminal interfaces. This structure is defined in the `ioctl.h` file, for which the `sgtty.h` file contains an `#include` statement. The `tchars` structure contains the following fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>t_intrc</td>
<td>The interrupt character (Ctrl-C, by default) generates a <code>SIGINT</code> signal. This is the normal way to stop a process that is no longer needed or to regain control in an interactive program.</td>
</tr>
<tr>
<td>t_quitc</td>
<td>The quit character (Ctrl-, by default) generates a <code>SIGQUIT</code> signal. This is used to end a program and produce a core image, if possible, in a <code>core</code> file in the current directory.</td>
</tr>
<tr>
<td>t_startc</td>
<td>The start-output character (Ctrl-Q, by default).</td>
</tr>
<tr>
<td>t_stopc</td>
<td>The stop-output character (Ctrl-S, by default).</td>
</tr>
<tr>
<td>t_eofc</td>
<td>The end-of-file character (Ctrl-D, by default).</td>
</tr>
<tr>
<td>t_brkc</td>
<td>The input delimiter (-1, by default). This character acts like a newline in that it ends a line, is echoed, and is passed to the program.</td>
</tr>
</tbody>
</table>

The stop and start characters can be the same to produce a toggle effect. The applicable `ioctl` functions are:

- **TIOCGETC**: Gets the special characters and puts them in the specified structure.
**TIOCSETC** Sets the special characters to those given in the structure.

**Local Mode**
Associated with each terminal is a local-mode word. The bits of the local-mode word are:

- **LCRTBS** Backspaces on erase rather than echoing erase.
- **LPRTERA** Printing terminal erase mode.
- **LCRTERA** Erases character echoes as Backspace-Space-Backspace.
- **LTILDE** Converts ~ (tilde) to ` (grave) on output (for terminals that do not recognize the tilde as an ASCII character).
- **LMDMBUF** Stops and starts output when carrier drops.
- **LLITOUT** Suppresses output translations.
- **LTOSTOP** Sends a **SIGTTOU** signal for background output.
- **LFLUSHO** Output is being flushed.
- **LNOHANG** Do not send hang up when carrier drops.
- **LCRTKIL** Backspace-Space-Backspace to erase the entire line on line kill.
- **LPASS8** Passes all 8 bits through on input, in any mode.
- **LCITLECH** Echoes input control characters as Ctrl-X, delete as Ctrl-?.
- **LPENDIN** Retypes pending input at next read or input character.
- **LDECCRTQ** Only Ctrl-Q restarts output after a Ctrl-S.
- **LNOFLSH** Inhibits flushing of pending I/O when an interrupt character is typed.

The following ioctl functions operate on the local-mode word structure:

- **TIOCLBIS** The *Value* parameter is a pointer to an integer whose value is a mask containing the bits to be set in the local-mode word.
- **TIOCLBIC** The *Value* parameter is a pointer to an integer whose value is a mask containing the bits to be cleared in the local-mode word.
- **TIOCLSET** The *Value* parameter is a pointer to an integer whose value is stored in the local-mode word.
- **TIOCLGET** The *Value* parameter is a pointer to an integer into which the current local-mode word is placed.

**Local Special Characters**
The *ltchars* structure associated with each terminal defines control characters for the new terminal driver. This structure contains the following fields:

- **t_suspc** The suspend-process character (Ctrl-Z, by default). This sends a **SIGTSTP** signal to suspend the current process group. This character is recognized during input.
- **t_dsuspc** The delayed suspend-process character (Ctrl-Y, by default). This sends a **SIGTSTP** signal to suspend the current process group. This character is recognized when the process attempts to read the control character rather than when the character is typed.
- **t_rprntc** The reprint line-control character (Ctrl-R, by default). This reprints all characters that are preceded by a new-line character and have not been read.
- **t_flushc** The flush-output character (Ctrl-O, by default). This flushes data that is written but not transmitted.
- **t_werasc** The word-erase character (Ctrl-W, by default). This erases the preceding word. This does not erase beyond the beginning of the line.
- **t_lnextc** The literal-next character (Ctrl-V, by default). This causes the special meaning of the next character to be ignored so that characters can be input without being interpreted by the system.

The following ioctl functions, which use the *ltchars* structure, are supported by the terminal interface for the definition of local special characters for a terminal:

- **TIOCSLTC** Sets local characters. The argument to this function is a pointer to an *ltchars* structure, which defines the new local special characters.
TIOCGLTC  Sets local characters. The argument to this function is a pointer to an ltchars structure into which is placed the current set of local special characters.

The winsize structure and the ioctl functions that use it are described in the discussion of the tty common code in "tty Subsystem Overview" in AIX 5L Version 5.3 General Programming Concepts: Writing and Debugging Programs.

File
/dev/tty  The tty special file, which is a synonym for the controlling terminal.

Related Information
The csh command, getty command, stty command, tset command.

The ioctl subroutine, sigvec subroutine.

shm.h File

Purpose
Describes the structures that are used by the subroutines that perform shared memory operations.

Syntax
#include <sys/shm.h>

Description
The shm.h header file defines the following symbolic constants, types, and structures:

Types:
typedef unsigned short  shmatt_t;

Symbolic Constants:
SHMLBA  segment low boundary address multiple
SHMLBA_EXTSHM  SHMLBA value when environment variable EXTSHM=ON
SHM_RDONLY  attach read-only (else read-write)
SHM_RND  round attach address to SHMLBA
SHM_MAP  map a file instead of share a segment
SHM_FMAP  fast file map
SHM_COPY  deferred update
SHM_R  read permission
SHM_W  write permission
SHM_DEST  destroy segment when # attached = 0
ZERO_MEM  for disclaim
SHMHISEG  highest shared memory segment allowed
SHMLOSEG  lowest shared memory segment allowed
NSHMSEGS  number of shared memory segments allowed

There is a shared mem id data structure for each shared memory and mapped file segment in the system.

Structures
The structure shmid_ds contains the following members:
The structure `shminfo` contains the following members:

- `unsigned int shmmmax`: max shared memory segment size
- `int shmmmin`: min shared memory segment size
- `int shmmni`: # of shared memory identifiers

The types `pid_t`, `time_t`, `key_t`, and `size_t` are defined as described in `<sys/types.h>`. The following are declared as functions:

- `void *shmat(int, const void *, int);`
- `int shmctl(int, int, struct shmid_ds *);`
- `int shmdt(const void *);`
- `int shmget(key_t, size_t, int);`

In addition, all of the symbols from `<sys/ipc.h>` will be defined when this header is included.

### Related Information

The `<types.h>` file.

The `shmat`, `shmctl`, `shmdt`, and `shmget` subroutines.

### spc.h File

#### Purpose

Defines external interfaces provided by the System Resource Controller (SRC) subroutines.

#### Description

The `/usr/include/spc.h` file defines data structures and symbolic constants that are used when calling the SRC subroutines. All subsystems that are controlled by the SRC via sockets or message queues should include this header file.

The `srcreq` data structure in the `spc.h` file defines the format of requests sent to a subsystem by the `srcmstr` daemon. This format is also used by SRC subroutines that send requests to the `srcmstr` daemon.

The `srcreq` data structure contains the following fields:

- `mtype`: The message type for the message queue. This field should be included only for message queue subsystems. Programs should be compiled with the `DSRCPYQUEUE` flag to generate the `mtype` field.
- `srchdr`: The SRC header that must be included in all packets sent to and received from an SRC subsystem.
- `subreq`: The request to be processed by the SRC subsystem.

The `srchdr` data structure in the `srcreq` data structure contains the return address that is needed to reply to the request. The `srcrrqs` subroutine can be used to extract this information from the request. The `srchdr` data structure is also part of the reply structure returned by a subsystem.
The `srchdr` data structure contains the following fields:

- **retaddr**: The return address
- **dversion**: The SRC packet version.
- **cont**: The continuation indicator. The possible values are:
  - **NEWREQUEST**: Used in a request to the `srcmstr` daemon.
  - **CONTINUED**: Used in a reply returned by a subsystem, indicating another packet follows.
  - **STATCONTINUED**: Used in a status reply returned by a subsystem, indicating another packet follows.
  - **END**: Used in a request seen by a subsystem or the last packet in reply sequence.

The `subreq` data structure contains the request to be processed by the subsystem. This same structure is used when calling the `srcsrqt` subroutine to send a request to a subsystem. The `srcsrqt` subroutine formats the required `srchdr` structure. The request is processed by the `srcmstr` daemon and passed on to a subsystem.

The `subreq` data structure contains the following fields:

- **object**: Defines the object on which to act. The possible values are either the `SUBSYSTEM` constant, or a subserver code point. If the object is a subsystem, the value of this field is the `SUBSYSTEM` constant as defined in the `spc.h` file and the `objname` field contains either a null value or the subsystem name. If the object is a subserver, the `object` field value is the code point from the subserver object definition, and the `objname` field is subsystem-defined. The `objname` field can be null, the subserver name, or the subserver process ID. The object value for the subserver cannot equal the value reserved for the subsystem.
- **action**: SRC action to perform. Possible types are:
  - **START**
  - **STOP**
  - **STATUS** or **SRCSTATUS**
  - **TRACE**
  - **REFRESH**

  The values 0-255 are reserved for use by the SRC.

- **parm1**: Modifies the SRC action type by indicating a variable associated with an action. This field is used in a different manner by each of the actions.
- **parm2**: Modifies the SRC action type by indicating a variable associated with an action. This field is used in a different manner by each of the actions.
- **objname**: Name of the object that the request applies to. This can be a subsystem name, a subserver object, or a subserver process ID.

The `srcrep` and `statrep` structures in the `spc.h` file define formats for the replies returned by a subsystem. For more information, see the `srcsrpy` subroutine.

The `srcrep` data structure must be used for replies to start, stop, refresh, and trace requests. It contains the following fields:

- **srchdr**: Specifies the SRC request/reply (srchdr) header.
- **svrreply**: A reply structure containing the following fields:
- **rtncode**: Subsystem response to the request. This response is negative on error or subsystem unique message.
The **statrep** data structure is used for replies to status requests. It contains the following fields:

- **srchdr**: Specifies the SRC request/reply (`srchdr`) header.
- **statcode**: A status structure containing the following fields. There may be an array of these structures. This structure contains the following fields:
  - **objtype**: The object type. This is one of the following:
    - **SUBSYSTEM**
    - Subserver code point
    - Error code
  - **status**: Status code. See the **spc.h** file for the symbolic constants that may be used with this field.
  - **objtext**: Text description.
  - **objname**: Name of the object (subsystem/subserver) this reply belongs to.

The **spc.h** file also defines the following constants that are useful in communicating with the **srcmstr** daemon:

- **SRCNAMESZ**: The maximum length of an SRC object name (30 bytes, including the null terminator).
- **SRCPKTMAX**: The maximum packet size (8192 bytes).

There are also SRC subroutines to manage SRC objects, including subsystems and subservers. The **spc.h** file defines certain symbolic constants which are useful when defining object attributes. The following SRC object descriptors are defined in the `/usr/include/sys/srcobj.h` file:

**Respawn action:**

- **RESPAWN=1**
- **ONCE=2**

**Contact options:**

- **SRCIPC=1**
- **SRCSIGNAL=2**
- **SRCSOCKET=3**

Multiple instances of a subsystem are allowed:

- **SRCYES=1**
SRCNO=0

Display subsystem status under certain conditions:

SRCYES=1

SRCNO=0

Default time limit:

TIMELIMIT=20 (seconds)

The spc.h file also includes the/usr/include/srcerrno.h file, which contains symbolic constants for the errors returned by the SRC library subroutines. The src_err_msg subroutine can be used to retrieve the corresponding error message.

**SRC Request Structure Example**

The following program excerpt is an example of the SRC request (srcreq) structure.

```c
struct srcreq
{
    long mtype; /*Contains the message type in the IPC buffer*/
    /*This field is included if IPC is used and a
     message queue is expected*/
    struct srchdr srchdr; /*src header table entry - defined below*/
    struct subreq subreq; /*the request passed to the subsystem*/
};

struct srchdr /*srchdr structure is used by SRC routines*/
    /*subsystems are not responsible for setting \this*/
{
    struct sockaddr_un retaddr;
    short dversion; /*the version of the data format*/
    short cont; /*used to indicate message is continued*/
};

struct subreq
{
    short object; /*object to act on*/
    short action; /*action START, STOP, STATUS,TRACE,REFRESH*/
    short parm1; /* */
    short parm2; /* */
    char objname[SRCNAMES]; /*object name*/
};
```

**Related Information**

The srcobj.h file.

The srcrq subroutine, srcsrpy subroutine, srcsrq subroutine, srcstat subroutine, srcstathdr subroutine, srcsbuf subroutine, srcstatxt subroutine, src_err_msg subroutine.

System Resource Controller (SRC) Overview for Programmers in AIX 5L Version 5.3 General Programming Concepts: Writing and Debugging Programs.
srcobj.h File

Purpose
Defines object structures used by the System Resource Controller (SRC) subsystem.

Description
The `/usr/include/sys/srcobj.h` header file contains the structures defining SRC objects. The `SRCsubsys` structure contains the following fields:

- **subsysname**: String that contains the subsystem name. This string can contain 30 bytes, including the null terminator.
- **synonym**: String that contains the subsystem synonym. This string can contain 30 bytes, including the null terminator.
- **cmdargs**: String that contains the subsystem command arguments. This string can contain 200 bytes, including the null terminator.
- **path**: String that contains the path to the executable files. This string can contain 200 bytes, including the null terminator.
- **uid**: User ID for the subsystem.
- **auditid**: Audit ID for the subsystem. This value is supplied by the system and cannot be changed by an SRC subroutine.
- **standin**: String that contains the path for standard input. This string can contain 200 bytes, including the null terminator.
- **standout**: String that contains the path for standard output. This string can contain 200 bytes, including the null terminator.
- **standerr**: String that contains the path for standard error. This string can contain 200 bytes, including the null terminator.
- **action**: Respawn action. The value of this field can be either `ONCE` or `RESPAWN`.
- **multi**: Multiple instance support. The value of this field can be either `SRCYES` or `SRCNO`.
- **contact**: Contact type. The value of this field indicates either a signal (`SRCSIGNAL`), a message queue (`SRCIPC`), or a socket (`SRCSOCKET`).
- **srvkey**: IPC message queue key.
- **svrmtype**: IPC message type (mtype) for the subsystem.
- **priority**: Nice value, a number from 1 to 40.
- **signorm**: Stop normal signal.
- **sigforce**: Stop force signal.
- **display**: Display inactive subsystem on all or group status. The value of this field can be either `SRCYES` or `SRCNO`.
- **waittime**: Stop cancel time to wait before sending a `SIGKILL` signal to the subsystem restart time period. (A subsystem can be restarted only twice in this time period if it does not terminate normally.
- **grpname**: String that contains the group name of the subsystem. This string can contain 30 bytes, including the null terminator.

The `SRCsubsvr` structure contains the following fields:

- **sub_type**: String that contains the type of the subsystem. This string can contain 30 bytes, including the null terminator.
- **subsysname**: String that contains the subsystem name. This string can contain 30 bytes, including the null terminator.
- **sub_code**: Subsystem code. This is a decimal number.

The `SRCnotify` structure contains the following fields:

- **notifyname**: String that contains the name of the subsystem or group to which the notify method applies. This string can contain 30 bytes, including the null terminator.
notifymethod

String that is executed when the SRC detects abnormal termination of the subsystem or group. This string can contain 256 bytes, including the null terminator.

The possible values indicated for the fields are predefined.

**Related Information**

The `spc.h` file.

The `getsys` subroutine.

Defining Your Subsystem to the SRC in *AIX 5L Version 5.3 General Programming Concepts: Writing and Debugging Programs*.

System Resource Controller (SRC) Overview for Programmers in *AIX 5L Version 5.3 General Programming Concepts: Writing and Debugging Programs*.

List of SRC Subroutines in *AIX 5L Version 5.3 General Programming Concepts: Writing and Debugging Programs*.

---

**stat.h File**

**Purpose**

Defines the data structures returned by the stat family of subroutines.

**Description**

The `stat` data structure in the `/usr/include/sys/stat.h` file returns information for the `stat`, `fstat`, `lstat`, `statx`, and `fstatx` subroutines.

The `stat` data structure contains the following fields:

- **st_dev**: Device that contains a directory entry for this file.
- **st_ino**: Index of this file on its device. A file is uniquely identified by specifying the device on which it resides and its index on that device.
- **st_mode**: File mode. The possible file mode values are given in the description of the `/usr/include/sys/mode.h` file.
- **st_nlink**: Number of hard links (alternate directory entries) to the file created using the `link` subroutine.
- **st_access**: Field is not implemented. All bits are returned as zero.
- **st_size**: Number of bytes in a file (including any holes). This field also defines the position of the end-of-file mark for the file. The end-of-file mark is updated only by subroutines, for example the `write` subroutine. If the file is mapped by the `shmat` subroutine and a value is stored into a page past the end-of-file mark, that mark will be updated to include this page when the file is closed or forced to permanent storage.
- **st_rdev**: ID of the device. This field is defined only for block or character special files.
- **st_atime**: Time when file data was last accessed. `st_atime` and `st_atime_n` taken together represent the last file access time in number of seconds and nanoseconds since the epoch.
- **st_atime_n**: Time when file data was last accessed. `st_atime` and `st_atime_n` taken together represent the last file access time in number of seconds and nanoseconds since the epoch.
- **st_mtime**: Time when file data was last modified. `st_mtime` and `st_mtime_n` taken together represent the last file modification time in number of seconds and nanoseconds since the epoch.
- **st_mtime_n**: Time when file data was last modified. `st_mtime` and `st_mtime_n` taken together represent the last file modification time in number of seconds and nanoseconds since the epoch.
- **st_ctime**: Time when the file status was last changed. `st_ctime` and `st_ctime_n` taken together represent the last file status change in number of seconds and nanoseconds since the epoch.
### st_ctime_n
Time when the file status was last changed. `st_ctime` and `st_ctime_n` taken together represent the last file status change in number of seconds and nanoseconds since the epoch.

### st_blksize
Size, in bytes of each block of the file.

### st_blocks
Number of blocks actually used by the file (measured in the units specified by the `DEV_BSIZE` constant).

### st_gen
Generation number of this i-node.

### st_type
Type of the v-node for the object. This is one of the following values, which are defined in the `/usr/include/sys/vnode.h` file:

- **VNON**: Unallocated object; this should not occur
- **VBAD**: Unknown type of object
- **VREG**: Regular file
- **VDIR**: Directory file
- **VBLK**: Block device
- **VCHR**: Character device
- **VLNK**: Symbolic link
- **VSOCK**: Socket
- **VFIFO**: FIFO
- **VMPC**: Multiplexed character device.

### st_vfs
Virtual file system (VFS) ID, which identifies the VFS that contains the file. By comparing this value with the VFS numbers returned by the `mntctl` subroutine, the name of the host where the file resides can be identified.

### st_vfstype
File-system type, as defined in the `/usr/include/sys/vmount.h` file.

### st_flag
Flag indicating whether the file or the directory is a virtual mount point. This flag can have the following values:

- **FS_VMP**: Indicates that the file is a virtual mount point.
- **FS_MOUNT**: Indicates that the file is a virtual mount point.
- **FS_REMOTE**: Indicates that the file resides on another machine.

### st_uid
File owner ID.

### st_gid
File group ID.

The **stat64** data structure in the `/usr/include/sys/stat.h` file returns information for the **stat64**, **fstat64**, and **lstat64** subroutines. The **stat64** structure contains the same fields as the **stat** structure, with the exception of the following field:

- **st_size**: Number of bytes in a file. The `st_size` field is a 64-bit quantity, allowing file sizes greater than `OFF_MAX`. The `st_size` field of the **stat64** structure is of the type `off64_t`.

For remote files, the **st_atime**, **st_mtime**, and **st_ctime** fields contain the time at the server.

The value of the `st_atime` field can be changed by the following subroutines:

- **read, readx, readv, readvx**
- **readlink**
- **shmdt**
- **utime, utimes**
The values of the `st_ctime` and `st_mtime` fields can be set by the following subroutines:

- `write`, `writex`, `writev`, `writevx`
- `open`, `openx`, `creat`
- `link`
- `symlink`
- `unlink`
- `mknod`
- `mkdir`
- `rename`
- `truncate`, `ftruncate`
- `utime`, `utimes`

In addition, the `shmdt` subroutine can change the `st_mtime` field, and the `chmod`, `fchmod`, `chown`, `chownx`, `fchown`, and `fchownx` subroutines can change the `st_ctime` field.

Because they can create a new object, the `open`, `openx`, `creat`, `symlink`, `mknod`, `mkdir`, and `pipe` subroutines can set the `st_atime`, `st_ctime`, and `st_mtime` fields.

**Related Information**

- The `mode.h` file, `types.h` file, `vmount.h` file.
- The `chmod` subroutine, `chownx` subroutine, `link` subroutine, `mknod or mkfifo` subroutine, `openx`, `open`, `creat` subroutine, `pipe` subroutine, `read` subroutine, `shmat` subroutine, `statx`, `stat`, `fstatx`, `fstat`, `fullstat`, or `ffullstat` subroutine, `unlink` subroutine, `utime` subroutine, `write`, `writex`, `writev`, or `writevx` subroutine.

---

**statfs.h File**

**Purpose**

Describes the structure of the statistics returned by the `statfs`, `fstatfs`, or `ustat` subroutine.

**Description**

The `statfs` and `fstatfs` subroutines return information on a mounted (virtual) file system in the form of a `statfs` structure. The `/usr/include/sys/statfs.h` file describes the `statfs` structure, which contains the following fields:

- `f_version` : Version number of the `statfs` structure. This value is currently 0.
- `f_length` : Length of the buffer that contains the returned information. This value is currently 0.
- `f_type` : Type of information returned. This value is currently 0.
- `f_bsize` : Optimal block size of the file system.
- `f_blocks` : Total number of blocks in the system.
- `f_bfree` : Number of free blocks in the file system. The size of a free block is given in the `f_bsize` field.
- `f_bavail` : Number of free blocks available to a nonroot user.
- `f_files` : Total number of file nodes in the file system.
- `f_ffree` : Number of free file nodes in the file system.
- `f_fsid` : File system ID.
f_vfstype  Type of this virtual file system. Possible values are:

MNT_JFS  Journaled File System (JFS) of the operating system

MNT_NFS  SUN network file system

MNT_CDROM  CD-ROM file system.

f_fsize  Fundamental block size of the file system.

f_fpack  File system pack name. The value returned by this field depends on the type of file system:

JFS  The value returned is copied from the s_fpack field of the superblock (see the `filsys.h` file format). You can set this value at the time the file system is created using the `mkfs` command with the `-v` flag.

CD-ROM  The value is copied from the volume identifier field in the primary volume descriptor.

NFS  The field is filled with null bytes because the f_fpack field is not implemented.

f_fpack

f_fsize

f_fname  File system name. The value returned by this field depends on the type of file system:

JFS  Value returned is copied from the s_fname field of the superblock (see the `filsys.h` file format). You can set this value at the time the file system is created by using the `mkfs` command with the `-l` flag. This field gives the preferred mount point for the file system.

Note: The s_fsize field in the superblock is only 6 bytes wide. Longer names are truncated to fit.

CD-ROM  The field is filled with null bytes because the f_fname field is not implemented.

NFS  The field is filled with null bytes because the f_fname field is not implemented.

f_fpack

f_fsize

f_fname

f_fpack

f_name_max  Maximum length of a component name for this file system.

Note: Fields that are not defined for a particular file system are set to a value of -1.

The `ustat` system returns information on a mounted file system in the form of a `ustat` structure. The `ustat` structure, which is defined in the `/usr/include/ustat.h` file, contains the following fields:

f_tfree  Total number of free blocks in the file system. The size of a free block is given in by the `UBSIZE` constant. See the `param.h` file for a description of `UBSIZE`.

f_inode  Number of free i-nodes in the file system.

f_fname  File system name.

f_fpack  File system pack name.

Files

`statfs.h`  Path to the `statfs.h` file.

`ustat.h`  Path to the `ustat.h` file.

Related Information

The `filsys.h` file format, `param.h` file, `vmount.h` file.

The `statfs`, `fstatfs`, or `ustat` subroutine.
statvfs.h File

Purpose
Describes the structure of the statistics that are returned by the statvfs subroutines and fsatvfs subroutines.

Description
The statvfs subroutines and fsatvfs subroutines return information on a mounted filesystem in the form of statvfs. The /usr/include/sys/statvfs.h file describes the following fields in the statvfs subroutine:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>f_bsize</td>
<td>Preferred file system block size</td>
</tr>
<tr>
<td>f_frsize</td>
<td>Fundamental file system block size</td>
</tr>
<tr>
<td>f_block</td>
<td>Total number of block f_frsize in the file system.</td>
</tr>
<tr>
<td>f_bfree</td>
<td>Total number of free blocks of f_frsize in the file system.</td>
</tr>
<tr>
<td>f_bavail</td>
<td>Total number of available blocks of f_frsize that can be used by users without root access.</td>
</tr>
<tr>
<td>f_files</td>
<td>Total number of file nodes in the file system.</td>
</tr>
<tr>
<td>f_ffree</td>
<td>Number of free file nodes in the file system.</td>
</tr>
<tr>
<td>f_favail</td>
<td>Number of free file nodes that can be used without root access.</td>
</tr>
<tr>
<td>f_fsid</td>
<td>File system ID.</td>
</tr>
<tr>
<td>f_basetype</td>
<td>File system type name</td>
</tr>
<tr>
<td>f_flag</td>
<td>File system flags:</td>
</tr>
<tr>
<td></td>
<td>ST_RDONLY File system is mounted read only</td>
</tr>
<tr>
<td></td>
<td>ST_NOSUID File system does not support set used ID file modes</td>
</tr>
<tr>
<td></td>
<td>ST_NODEV Device opens are not allowed through mounts.</td>
</tr>
<tr>
<td>f_namemax</td>
<td>Maximum length of a component name for this file system</td>
</tr>
<tr>
<td>f_fstr</td>
<td>File system specific string.</td>
</tr>
</tbody>
</table>

The following prototypes also appear in the /usr/include/sys/statvfs.h file:
extern int statvfs(const char *, struct statvfs *);
extern int fsatvfs(int, struct statvfs *);

Related Information
The ststvfs subroutine, fstatvfs subroutine.

systemcfg.h File

Purpose
Defines the _system_configuration structure.

Description
The systemcfg.h file defines the _system_configuration structure. This is a global structure that identifies system characteristics. The _system_configuration structure is provided in read-only system memory. New fields will be added to the structure in future releases. The attributes in the _system_configuration structure have the following values:
architecture Identifies the processor architecture. Valid values for Version 4 are:

POWER_RS
  Indicates a POWER family machine.

POWER_PC
  Indicates a POWER-based.

implementation Identifies the specific version of the processor. Each implementation is assigned a unique bit to allow for efficient checking of implementation sets. The following are examples of valid values (the header file contains more values):

POWER_RS1
POWER_RS2
POWER_RSC
POWER_601

Two special values are also defined: POWER_RS_ALL and POWER_PC_ALL. These labels are defined as the bit OR of all members of their architecture.

version Identifies the central processing unit (CPU) version number. The following are examples of valid values (the header file contains more values):

PV_RS1
  Identifies a POWER family RS1 machine.
PV_RS2
  Identifies a POWER family RS2 machine.
PV_RS2G
  Identifies a POWER family RS2 machine with graphics assist.
PV_RSC
  Identifies a POWER family RSC machine.
PV_601
  Identifies a PowerPC 601 RISC Microprocessor machine.

width Contains the processor data-word size. Valid values are 32 or 64. This value is the maximum data-word size and should not be confused with the current execution mode.

ncpus Identifies the number of CPUs active on a machine. Uniprocessor (UP) machines are identified by a 1. Values greater than 1 indicate multiprocessor (MP) machines.

cache_attr Specifies the cache attributes. Bit 31 determines if the cache is present. If this bit is 1, the cache is present. If bit 31 is 0, then no cache is present and all other cache characteristics are 0. Bit 30 indicates the type of cache. If bit 30 is 1, the cache is combined. Otherwise, if bit 30 is 0 the instruction and data caches are separate.

icache_size Contains the L1 instruction-cache size in bytes. For combined caches, this value is the total cache size.

dcache_size Contains the size of the L1 data-cache size in bytes. For combined caches this the total cache size.

icache_asc Contains the L1 instruction-cache associativity. For a combined cache, this is the combined caches' associativity.

dcache_asc Contains the L1 data-cache associativity. For a combined cache, this is the combined caches' associativity.

icache_line Contains the line size in bytes of the L1 instruction cache.

dcache_line Contains the line size in bytes of L1 data cache.

L2_cache_size Contains the size of the L2 cache in bytes. A value of 0 indicates no L2 cache is present.

L2_cache_asc Identifies the associativity of the L2 cache.

tlb_comb Identifies the type of Transaction Lookaside Buffer (TLB) attributes. If the TLB is present, bit 31 is 1. Otherwise, if bit 31 is less than 0, the TLB does not exist and all other TLB characteristics are 0. Bit 30 is 1 if the TLB is combined. If the TLB is separate for the instruction and data cache, bit 30 is 0.

itlb_size Specifies the number of entries in the instruction TLB. For combined TLBs, this is the size of the combined TLB.
dtlb_size
Specifies the number of entries in the data TLB. For combined TLBs, this is the size of
the combined TLB.

itlb_asc
Contains the associativity of the instruction TLB. This attribute’s value is equal to the
itlb_size attribute if the system is fully associative.

dtlb_asc
Contains the associativity of the instruction TLB. This attribute’s value is equal to the
value of the dtlb_size attribute if the system is fully associative.

resv_size
Contains the POWER-based reservation granule size. This field is a 0 on POWER family
machines.

priv_ick_cnt
Contains the number of times lock services attempt to lock a spin lock before blocking AP
process/thread in supervisor mode. This a 0 on UP machine. This parameter is used by
system-locking services.

prob_lck_cnt
Contains the number of times lock services attempt to lock a spin lock before blocking a
process or thread in problem state. This a 0 on a UP machine. This parameter is used by
system-locking services.

virt_alias
Indicates virtual memory aliasing. If 1, the hardware is available for virtual memory
aliasing and this ability is used by the system. Virtual memory aliasing is the mapping of
one real address to more than one virtual address.

cach_cong
Contains the number page index bits that can result in a cache synonym. For machines
without cache synonyms, this field is 0.

tar.h File

Purpose
Contains definitions for flags used in the tar archive header.

Description
The /usr/include/tar.h file contains extended definitions used in the typeflag and mode fields of the tar
archive header block. The file also provides values for the required POSIX entries.

tar Archive Header Block
Every file archived using the tar command is represented by a header block describing the file, followed
by zero or more blocks that give the contents of the file. The end-of-archive indicator consists of two
blocks filled with binary zeros. Each block is a fixed size of 512 bytes.

Blocks are grouped for physical I/O operations and groups can be written using a single write subroutine
operation. On magnetic tape, the result of this write operation is a single tape record. The last record is
always a full 512 bytes. Blocks after the end-of-archive zeros contain undefined data.

The header block structure is shown in the following table. All lengths and offsets are in decimal.

Table 36. Header Block Structure

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Offset: 0</td>
</tr>
<tr>
<td></td>
<td>Length in Bytes: 100</td>
</tr>
<tr>
<td></td>
<td>Contents: File name without a / (slash)</td>
</tr>
<tr>
<td>Field Name</td>
<td>Structure</td>
</tr>
<tr>
<td>------------</td>
<td>-----------</td>
</tr>
</tbody>
</table>
| mode       | Offset: 100  
Length in Bytes: 8  
Contents: File mode |
| uid        | Offset: 108  
Length in Bytes: MAXIMPL_LOGIN_NAME_MAX  
Contents: User ID |
| gid        | Offset: 116  
Length in Bytes: MAXIMPL_LOGIN_NAME_MAX  
Contents: Group ID |
| size       | Offset: 124  
Length in Bytes: 12  
Contents: Size in bytes |
| mtime      | Offset: 136  
Length in Bytes: 12  
Contents: Latest modification time |
| cksum      | Offset: 148  
Length in Bytes: 8  
Contents: File and header checksum |
| typeflag   | Offset: 156  
Length in Bytes: 1  
Contents: File type |
| linkname   | Offset: 157  
Length in Bytes: 100  
Contents: Linked path name or file name |
<table>
<thead>
<tr>
<th>Field Name</th>
<th>Structure</th>
</tr>
</thead>
</table>
| magic      | Offset: 257  
              Length in Bytes: 6  
              Contents: Format representation for tar |
| version    | Offset: 263  
              Length in Bytes: 3  
              Contents: Version representation for tar |
| uname      | Offset: 265  
              Length in Bytes: 32  
              Contents: User name |
| gname      | Offset: 297  
              Length in Bytes: 32  
              Contents: Group name |
| devmajor   | Offset: 329  
              Length in Bytes: 8  
              Contents: Major device representation |
| devminor   | Offset: 337  
              Length in Bytes: 8  
              Contents: Minor device representation |
| prefix     | Offset: 345  
              Length in Bytes: 155  
              Contents: Path name without trailing slashes |

Names are preserved only if the characters are chosen from the POSIX portable file-name character set or if the same extended character set is used between systems. During a read operation, a file can be created only if the original file can be accessed using the `open`, `stat`, `chdir`, `fcntl`, or `opendir` subroutine.
Header Block Fields
Each field within the header block and each character on the archive medium are contiguous. There is no
padding between fields. More information about the specific fields and their values follows:

name  The file's path name is created using this field, or by using this field in connection with the prefix
field. If the prefix field is included, the name of the file is prefix/name. This field is null-terminated
unless every character is non-null.

mode  Provides 9 bits for file permissions and 3 bits for SUID, SGID, and SVTX modes. All values for this
field are in octal. During a read operation, the designated mode bits are ignored if the user does
not have equal (or higher) permissions or if the modes are not supported. Numeric fields are
terminated with a space and a null byte. The tar.h file contains the following possible values for
this field:

<table>
<thead>
<tr>
<th>Flag</th>
<th>Octal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSUID</td>
<td>04000</td>
<td>Set user ID on execution.</td>
</tr>
<tr>
<td>TSGID</td>
<td>02000</td>
<td>Set group ID on execution.</td>
</tr>
<tr>
<td>TSVTX</td>
<td>01000</td>
<td>Reserved.</td>
</tr>
<tr>
<td>TUREAD</td>
<td>00400</td>
<td>Read by owner.</td>
</tr>
<tr>
<td>TUWRITE</td>
<td>00200</td>
<td>Write by owner.</td>
</tr>
<tr>
<td>TUEXEC</td>
<td>00100</td>
<td>Execute or search by owner.</td>
</tr>
<tr>
<td>TGREAD</td>
<td>00040</td>
<td>Read by group.</td>
</tr>
<tr>
<td>TGWRITE</td>
<td>00020</td>
<td>Write by group.</td>
</tr>
<tr>
<td>TGEXEC</td>
<td>00010</td>
<td>Execute or search by group.</td>
</tr>
<tr>
<td>TOREAD</td>
<td>00004</td>
<td>Read by others.</td>
</tr>
<tr>
<td>TOWRITE</td>
<td>00002</td>
<td>Write by others.</td>
</tr>
<tr>
<td>TOEXEC</td>
<td>00001</td>
<td>Execute or search by other.</td>
</tr>
</tbody>
</table>

uid    Extracted from the corresponding archive fields unless a user with appropriate privileges restores
the file. In that case, the field value is extracted from the password and group files instead.
Numeric fields are terminated with a space and a null byte.

gid    Extracted from the corresponding archive fields unless a user with appropriate privileges restores
the file. In that case, the field value is extracted from the password and group files instead.
Numeric fields are terminated with a space and a null byte.

size   Value is 0 when the typeflag field is set to LNKTYPE. This field is terminated with a space only.

mtime  Value is obtained from the modification-time field of the stat subroutine. This field is terminated
with a space only.

chksum On calculation, the sum of all bytes in the header structure are treated as spaces. Each unsigned
byte is added to an unsigned integer (initialized to 0) with at least 17-bits precision. Numeric fields
are terminated with a space and a null byte.

typeflag The tar.h file contains the following possible values for this field:

<table>
<thead>
<tr>
<th>Flag</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REGTYPE</td>
<td>'0'</td>
<td>Regular file.</td>
</tr>
<tr>
<td>AREGTYPE</td>
<td>'0'</td>
<td>Regular file.</td>
</tr>
<tr>
<td>LNKTYPE</td>
<td>'1'</td>
<td>Link.</td>
</tr>
<tr>
<td>SYMTYPE</td>
<td>'2'</td>
<td>Reserved.</td>
</tr>
<tr>
<td>Flag</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CHRTYPE</td>
<td>‘3’</td>
<td>Character special.</td>
</tr>
<tr>
<td>BLKTYPE</td>
<td>‘4’</td>
<td>Block special.</td>
</tr>
<tr>
<td>DIRTYPE</td>
<td>‘5’</td>
<td>Directory. In this case, the size field has no meaning.</td>
</tr>
<tr>
<td>FIFOTYPE</td>
<td>‘6’</td>
<td>FIFO special. Archiving a FIFO file archives its existence, not contents.</td>
</tr>
<tr>
<td>CONNTYPE</td>
<td>‘7’</td>
<td>Reserved.</td>
</tr>
</tbody>
</table>

If other values are used, the file is extracted as a regular file and a warning issued to the standard error output. Numeric fields are terminated with a space and a null byte.

The **LNKTYPE** flag represents a link to another file, of any type, previously archived. Such linked-to files are identified by each file having the same device and file serial number. The linked-to name is specified in the **linkname** field, including a trailing null byte.

**linkname**

Does not use the **prefix** field to produce a path name. If the path name or **linkname** value is too long, an error message is returned and any action on that file or directory is canceled. This field is null-terminated unless every character is non-null.

**magic**

Contains the **TMAGIC** value, reflecting the extended tar archive format. In this case, the **uname** and **gname** fields will contain the ASCII representation for the file owner and the file group. If a file is restored by a user with the appropriate privileges, the **uid** and **gid** fields are extracted from the password and group files (instead of the corresponding archive fields). This field is null-terminated.

**version**

Represents the version of the **tar** command used to archive the file. This field is terminated with a space only.

**uname**

Contains the ASCII representation of the file owner. This field is null-terminated.

**gname**

Contains the ASCII representation of the file group. This field is null-terminated.

**devmajor**

Contains the device major number. Terminated with a space and a null byte.

**devminor**

Contains the device minor number. Terminated with a space and a null byte.

**prefix**

If this field is non-null, the file’s path name is created using the **prefix/name** values together. Null-terminated unless every character is non-null.

### Related Information

The **tar** command.

### termio.h File

#### Purpose

Defines the structure of the **termio** file, which provides the terminal interface for Version 2 compatibility.

#### Description

The **/usr/include/sys/termio.h** file contains the **termio** structure, which defines special characters as well as the basic input, output, control, and line discipline modes. The **termio.h** file is provided for compatibility with Version 2 applications.
Version 2 applications that include the `termio.h` file can use the Version 2 terminal interface provided by the POSIX line discipline. The following Version 2 terminal interface operations are not supported by the POSIX line discipline:

- Terminal Paging (`TCGLEN ioctl` and `TCSLEN ioctl`)
- Terminal Logging (`TCLOG ioctl`)
- Enhanced Edit Line Discipline (`LDSETDT ioctl` and `LDGETDT ioctl`)

The `termio` structure in the `termio.h` file contains the following fields:

- `c_iflag`
- `c_oflag`
- `c_cflag`
- `c_lflag`
- `c_cc`

**Field Descriptions**

**c_iflag**

Describes the basic terminal input control. The initial input-control value is all bits clear. The possible input modes are:

- **IGNBRK**
  
  Ignores the break condition. In the context of asynchronous serial data transmission, a break condition is defined as a sequence of zero-valued bits that continues for more than the time required to send 1 byte. The entire sequence of zero-valued bits is interpreted as a single break condition, even if it continues for an amount of time equivalent to more than one byte. If the IGNBRK flag is set, a break condition detected on input is ignored, which means that the break condition is not put on the input queue and therefore not read by any process.

- **BRKINT**
  
  Interrupts the signal on the break condition. If the IGNBRK flag is not set and the BRKINT flag is set, the break condition flushes the input and output queues. If the terminal is the controlling terminal of a foreground process group, the break condition generates a single SIGINT signal to that foreground process group. If neither the IGNBRK nor the BRKINT flag is set, a break condition is read as a single \0. If the PARMRK flag is set, a break condition is read as \377, \0, \0.

- **IGNPAR**
  
  Ignores characters with parity errors. If this flag is set, a byte with a framing or parity error (other than break) is ignored.

- **PARMRK**
  
  Marks parity errors. If the PARMRK flag is set and the IGNPAR flag is not set, a byte with a framing or parity error (other than break) is given to the application as the three-character sequence \377, \0, x, where \377, \0 is a two-character flag preceding each sequence and x is the data of the character received in error. To avoid ambiguity in this case, if the ISTRIP flag is not set, a valid character of \377 is given to the application as \377, \377. If neither the IGNPAR nor the PARMRK flag is set, a framing or parity error (other than break) is given to the application as a single character, \0.

- **INPCK**
  
  Enables input parity checking. If this flag is set, input parity checking is enabled. If not set, input parity checking is disabled. This allows for output parity generation without input parity errors.

- **ISTRIP**
  
  Strips characters. If this flag is set, valid input characters are first stripped to 7 bits; otherwise, all 8 bits are processed.
INLCR  
Maps a new-line character (NL) to a carriage return (CR) on input. If this flag is set, a received NL character is translated into a CR character.

IGNCR  
Ignores a CR character. If this flag is set, a received CR character is ignored and not read.

ICRNL  
Maps a CR character to an NL character on input. If the ICRNL flag is set and the IGNCR flag is not set, a received CR character is translated into an NL character.

IUCLC  
Maps uppercase to lowercase on input. If this flag is set, a received uppercase, alphabetic character is translated into the corresponding lowercase character.

IXON  
Enables start and stop output control. If this flag is set, a received STOP character suspends output and a received START character restarts output. When the IXON flag is set, START and STOP characters are not read, but merely perform flow-control functions. When the IXON flag is not set, the START and STOP characters are read.

IXANY  
Enables any character to restart output. If this flag is set, any input character restarts output that was suspended.

IXOFF  
Enables start-and-stop input control. If this flag is set, the system transmits a STOP character when the input queue is nearly full and a START character when enough input has been read that the queue is nearly empty again.

c_oflag  
Specifies how the system treats output. The initial output-control value is “all bits clear”. The possible output modes are:

OPOST  
Post processes output. If this flag is set, output characters are post-processed as indicated by the remaining flags; otherwise, characters are transmitted without change.

OLCUC  
Maps lowercase to uppercase on output. If this flag is set, a lowercase alphabetic character is transmitted as the corresponding uppercase character. This function is often used in conjunction with the IUCLC input mode.

ONLCR  
Maps NL to CR-NL on output. If this flag is set, the NL character is transmitted as the CR-NL character pair.

OCRNL  
Maps CR to NL on output. If this flag is set, the CR character is transmitted as the NL character.

ONOCR  
Indicates no CR output at column 0 (first position). If this flag is set, no CR character is transmitted when at column 0 (first position).

ONLRET  
NL performs the CR function. If this flag is set, the NL character is assumed to do the carriage-return function. The column pointer is set to 0, and the delay specified for carriage return is used. If neither the ONLCR, OCRNL, ONOCR, nor ONLRET flag is set, the NL character is assumed to do the line-feed function only. The column pointer remains unchanged. The column pointer is also set to 0 if the CR character is actually transmitted.

The delay bits specify how long a transmission stops to allow for mechanical or other movement when certain characters are sent to the terminal. The actual delays depend on line speed and system load.
**OFILL** Uses fill characters for delay. If this flag is set, fill characters are transmitted for a delay instead of a timed delay. This is useful for high baud rate terminals that need only a minimal delay.

**OFDEL**
If this flag is set, the fill character is DEL. If this flag is not set, the fill character is NULL.

**NLDLY**
Selects the new-line character delays. This is the mask to use before comparing to NL0 and NL1:
- **NL0** Specifies no delay.
- **NL1** Specifies one delay of approximately 0.10 seconds. If the **ONLRET** flag is set, the carriage-return delays are used instead of the new-line delays. If the **OFILL** flag is set, two fill characters are transmitted.

**CRDLY**
Selects the carriage-return delays. This is the mask to use before comparing to CR0, CR1, CR2, and CR3:
- **CR0** Specifies no delay.
- **CR1** Specifies that the delay is dependent on the current column position. If the **OFILL** flag is set, two fill characters are transmitted.
- **CR2** Specifies a delay of approximately 0.10 seconds. If the **OFILL** flag is set, this delay transmits four fill characters.
- **CR3** Specifies one delay of approximately 0.15 seconds.

**TABDLY**
Selects the horizontal-tab delays. This is the mask to use before comparing to TAB0, TAB1, TAB2, and TAB3. If the **OFILL** flag is set, any of these delays (except TAB3) transmit two fill characters:
- **TAB0** Specifies no delay.
- **TAB1** Specifies that the delay is dependent on the current column position. If the **OFILL** flag is set, two fill characters are transmitted.
- **TAB2** Specifies a delay of approximately 0.10 seconds.
- **TAB3** Specifies that tabs are to be expanded into spaces.

**BSDLY**
Selects the backspace delays. This is the mask to use before comparing to BS0 and BS1:
- **BS0** Specifies no delay.
- **BS1** Specifies a delay of approximately 0.05 seconds. If the **OFILL** flag is set, this delay transmits one fill character.

**VTDLY**
Selects the vertical-tab delays. This is the mask to use before comparing to VT0 and VT1:
- **VT0** Specifies no delay.
- **VT1** Specifies one delay of approximately 2 seconds.

**FFDLY**
Selects the form-feed delays. This is a mask to use before comparing to FF0 and FF1:
- **FF0** Specifies no delay.
- **FF1** Specifies a delay of approximately 2 seconds.
c_cflag

Describes the hardware control of the terminal. In addition to the basic control modes, this field uses the following control characters:

CBAUD

Specifies baud rate. These bits specify the baud rate for a connection. For any particular hardware, impossible speed changes are ignored.

B0  Specifies a zero baud rate which is used to hang up the connection. If B0 is specified, the 'data terminal ready' signal is not asserted. As a result, the line is usually disconnected. This delay transmits two fill characters. Normally, this disconnects the line.

B50  Specifies 50 baud.

B75  Specifies 75 baud.

B110  Specifies 110 baud.

B134  Specifies 134.5 baud.

B150  Specifies 150 baud.

B200  Specifies 200 baud.

B300  Specifies 300 baud.

B600  Specifies 600 baud.

B1200  Specifies 1200 baud.

B1800  Specifies 1800 baud.

B2400  Specifies 2400 baud.

B4800  Specifies 4800 baud.

B9600  Specifies 9600 baud.

B19200  Specifies 19,200 baud.

B38400  Specifies 38,400 baud.

EXTA  Specifies External A.

EXTB  Specifies External B.

CSIZE  Specifies the character size. These bits specify the character size, in bits, for both transmit and receive operations. The character size does not include the parity bit, if one is used:

CS5  5 bits

CS6  6 bits

CS7  7 bits

CS8  8 bits

CSTOPB  Specifies the number of stop bits. If this flag is set, 2 stop bits are sent; otherwise, only 1 stop bit is sent.

CREAD  Enables the receiver. If this flag is set, the receiver is enabled. Otherwise, characters are not received.
PARENB
Enables parity. If this flag is set, parity generation and detection is enabled and a parity bit is added to each character.

PARODD
Specifies odd parity. If parity is enabled, the PARODD flag specifies odd parity if set. If parity is enabled and the PARODD flag is not set, even parity is used.

HUPCL
Hangs up on last close. If this flag is set, the line is disconnected when the last process closes the line or when the process terminates (when the `data terminal ready' signal drops).

CLOCAL
Specifies a local line. If this flag is set, the line is assumed to have a local, direct connection with no modem control. If not set, modem control (dial-up) is assumed.

c_lflag
Controls various terminal functions. The initial value after an open is "all bits clear." This field uses the following mask name symbols:

ISIG  Enables signals. If this flag is set, each input character is checked against the INTR and QUIT special control characters. If an input character matches one of these control characters, the function associated with that character is performed. If the ISIG function is not set, checking is not done.

ICANON  Enables canonical input. If this flag is set, it turns on canonical processing, which enables the erase and kill edit functions as well as the assembly of input characters into lines delimited by NL, EOF, and EOL characters. If the ICANON flag is not set, read requests are satisfied directly from the input queue. In this case, a read request is not satisfied until one of the following conditions is met:
• The minimum number of characters specified by the MIN value are received.
• The time-out value specified by the TIME value has expired since the last character was received.

As a result bursts of input can be read, while still allowing single-character input. The MIN and TIME values are stored in the positions for the EOF and EOL characters, respectively. The character values of MIN and TIME are converted to their ascii equivalents to get the numeric value. The time value represents tenths of seconds.

XCASE
Enables canonical uppercase and lowercase presentation. If this flag is set along with the ICANON flag, an uppercase letter (or the uppercase letter translated to lowercase by the IUCLC input mode) is accepted on input by preceding it with a \ (backslash) character. The output is then also preceded by a backslash character. In this mode, the output generates and the input accepts the following escape sequences:

<table>
<thead>
<tr>
<th>For:</th>
<th>Use:</th>
</tr>
</thead>
<tbody>
<tr>
<td>` (grave)</td>
<td>\ `</td>
</tr>
<tr>
<td></td>
<td>\ !</td>
</tr>
<tr>
<td>~</td>
<td>\ ^</td>
</tr>
<tr>
<td>{</td>
<td>\ (</td>
</tr>
<tr>
<td>}</td>
<td>\ )</td>
</tr>
<tr>
<td>\</td>
<td>\ \</td>
</tr>
</tbody>
</table>

For example, A is input as \a, \n as \n, and \N as \n.
NOFLSH
Disables queue flushing. If this flag is set, the normal flushing of the input and output queues associated with the INTR and QUIT characters is not done.

ECHO Enables echo. If this flag is set, characters are echoed as they are received.
When the ICANON flag is set, the following echo functions are possible:

ECHOE Echoes the erase character as Backspace-Space-Backspace. If the ECHO and ECHOE flags are both set, the ERASE character is echoed as one or more ASCII Backspace-Space-Backspace sequences, which clears the last characters from the screen.

ECHOK Echoes the NL character after kill. If the ECHOK flag is set, the NL character is echoed after the kill character is received. This emphasizes that the line is deleted.

ECHONL Echoes the NL character. If the ECHONL flag is set, the NL character is echoed even if the ECHO flag is not set. This is useful for terminals that are set to “local echo” (also referred to as “half-duplex”).

c_cc Specifies an array that defines the special control characters. The relative positions and initial values for each function are:

VINTR Indexes the INTR special character (Ctrl-c), which is recognized on input if the ISIG flag is set. The INTR character generates a SIGINT signal, which is sent to all processes in the foreground process group for which the terminal is the controlling terminal. If the ISIG flag is set, the INTR character is discarded when processed.

VQUIT Indexes the QUIT special character (Ctrl-\), which is recognized on input if the ISIG flag is set. The QUIT character generates a SIGQUIT signal, which is sent to all processes in the foreground process group for which the terminal is the controlling terminal, and writes a core image file into the current working directory. If the ISIG flag is set, the QUIT character is discarded when processed.

VERASE Indexes the ERASE special character (Backspace), which is recognized on input if the ICANON flag is set. The ERASE character does not erase beyond the beginning of the line as delimited by a NL, EOL, EOF, or EOL2 character. If the ICANON flag is set, the ERASE character is discarded when processed.

VKILL Indexes the KILL special character (Ctrl-u), which is recognized on input if the ICANON flag is set. The KILL character deletes the entire line, as delimited by an NL, EOL, EOF, or EOL2 character. If the ICANON flag is set, the KILL character is discarded when processed.

VEOF Indexes the EOF special character (Ctrl-d), which is recognized on input if the ICANON flag is set. When EOF is received, all the characters waiting to be read are immediately passed to the process, without waiting for a new line, and the EOF is discarded. If the EOF is received at the beginning of a line (no characters are waiting), a character count of zero is returned from the read, indicating an end-of-file. If the ICANON flag is set, the EOF character is discarded when processed.

VEOL Indexes the EOL special character (Ctrl-@ or ASCII NULL), which is recognized on input if the ICANON flag is set. EOL is an additional line delimiter, like NL, and is not normally used.
VEOL2
Indexes the EOL2 special character (Ctrl-@ or ASCII NULL), which is recognized on input if the ICANON flag is set. EOL2 is another additional line delimiter, like NL, and is not normally used.

VMIN
Indexes the MIN value, which is not a special character. The use of the MIN value is described in the discussion of non-canonical mode input processing in "POSIX (termios.h File) Line Discipline" in AIX 5L Version 5.3 General Programming Concepts: Writing and Debugging Programs.

VTIME
Indexes the TIME value, which is not a special character. The use of the TIME value is described in the discussion of non-canonical mode input processing in "POSIX (termios.h File) Line Discipline" in AIX 5L Version 5.3 General Programming Concepts: Writing and Debugging Programs.

The character values for the following control characters can be changed:
- INTR
- ERASE
- EOF
- EOL2
- QUIT
- KILL
- EOL

The ERASE, KILL, and EOF characters can also be escaped (preceded with a backslash) so that no special processing is done.

The primary ioctl subroutines have the form:
ioctl (FileDescriptor, Command, Structure)
struct termio *Structure;

The operations using this form are:

TCGETA
Gets the parameters associated with the terminal and stores them in the termio structure referenced by the Structure parameter.

TCSETA
Sets the parameters associated with the terminal from the structure referenced by the Structure parameter. The change is immediate.

TCSETAF
Waits for the output to drain, and then flushes the input queue and sets the new parameters.

TCSETAW
Waits for the output to drain before setting the new parameters. This form should be used when changing parameters that will affect output.

Other ioctl subroutines have the form:
ioctl (FileDescriptor, Command, Value)
int Value;

The operations using this form are:

Note: If the user writes an application that performs a TCSBRK operation followed by a TCFLSH operation prior to closing a port, the last data left in the concentrator box on the 64-port adapter is lost. However, no problem occurs if an SIO, 8-port, or 16-port adapter is used.
TCSBRK
Waits for the output to drain. If the Value parameter has a value of 0, it sends a break of 0.25 seconds. A nonzero value causes a break condition of that many milliseconds.

TCSBREAK
Waits for the output to drain. If the Value parameter has a value of 0, it sends a break of .25 seconds. A nonzero value causes a break condition of that many milliseconds.

TCXONC
Starts and stops control. If the Value parameter has a value of 0, it suspends output. If the Value parameter has a value of 1, it restarts suspended output. If the Value parameter has a value of 2, it blocks input. If the Value parameter has a value of 3, it unblocks input.

TCFLSH
If the Value parameter has a value of 0, it flushes the input queue. If the Value parameter has a value of 1, it flushes the output queue. If the Value parameter has a value of 2, it flushes both the input and output queues.

Another form for ioctl subroutines is:
ioctl (FileDescriptor, Command, Structure)
struct csmap* Structure;

TCSCSMAP
Sets the code set map from the structure referenced by the structure parameter and rejects any invalid map (any map with 0 length/width or a length greater than MB_LEN_MAX). The /usr/include/sys/tty.h file contains the structure used for TCSCSMAP and TCGCSMAP operations.

TCGCSMAP
Returns a copy of the current code set map in the structure referenced by the structure parameter. The /usr/include/sys/tty.h file contains the structure used for TCSCSMAP and TCGCSMAP operations.

The following ioctl operations are used for trusted communications path operations:

TCSAK
Points to an integer that enables the Secure Attention Key (SAK) sequence (Ctrl-X, Ctrl-R) to provide a clean terminal to which only trusted processes can read or write. When SAK is enabled and the user types this sequence, all processes that are currently running are ended. The TCSAKON operation turns the SAK sequence on; the TCSAKOFF operation turns the SAK sequence off.

TCQSAK
Queries the state (TCSAKON or TCSAKOFF) of the SAK sequence.

TCTRUST
Sets a bit by which another process can query, (with the TCQTRUST operation), the state of the terminal, (TCTRUSTED or TCUNTRUSTED).

TCQTRUST
Queries the state of the terminal (TCTRUSTED or TCUNTRUSTED).

Related Information
The fork subroutine, ioctl subroutine, setpgrp subroutine, sigvec subroutine.

The csh command, getty command, stty command, tset command.

The tty Subsystem Overview in AIX 5L Version 5.3 General Programming Concepts: Writing and Debugging Programs.
termios.h File

Purpose
Defines the structure of the termios file, which provides the terminal interface for POSIX compatibility.

Description
The /usr/include/termios.h file contains information used by subroutines that apply to terminal files. The definitions, values, and structures in this file are required for compatibility with the POSIX standard. The termios.h file also supports ioctl modem-control operations.

The general terminal interface information is contained in the termio.h file. The termio structure in the termio.h file defines the basic input, output, control, and line discipline modes. If a calling program is identified as requiring POSIX compatibility, the termios structure and additional POSIX control-packet information in the termios.h file is implemented. Window and terminal size operations use the winsize structure, which is defined in the ioctl.h file. The termios structure in the termios.h file contains the following fields:

- c_iflag
- c_oflag
- c_cflag
- c_lflag
- c_cc

The termios.h file also defines the values for the following parameters of the tcsetattr subroutine:

- OptionalActions
- QueueSelector
- Action

The termios.h file also supports ioctl modem-control operations.

Field Descriptions

c_iflag
Describes the basic terminal input control. The initial input-control value is all bits clear. The possible input modes are:

IGNBRK
Ignores the break condition. In the context of asynchronous serial data transmission, a break condition is defined as a sequence of zero-valued bits that continues for more than the time required to send one byte. The entire sequence of zero-valued bits is interpreted as a single break condition, even if it continues for an amount of time equivalent to more than one byte. If the IGNBRK flag is set, a break condition detected on input is ignored, which means that it is not put on the input queue and therefore not read by any process.

BRKINT
Signal interrupt on the break condition. If the IGNBRK flag is not set and the BRKINT flag is set, the break condition flushes the input and output queues. If the terminal is the controlling terminal of a foreground process group, the break condition generates a SIGINT signal to that foreground process group. If neither the IGNBRK nor the BRKINT flag is set, a break condition is read as a single \0, or if the PARMRK flag is set, as \377, \0, \0.

IGNPAR
Ignores characters with parity errors. If this flag is set, a byte with a framing or parity error (other than break) is ignored.
PARMRK
Marks parity errors. If the PARMRK flag is set, and the IGNPAR flag is not set, a byte with a framing or parity error (other than break) is given to the application as the three-character sequence \377, \0, x, where \377, \0 is a two-character flag preceding each sequence and x is the data of the character received in error. To avoid ambiguity in this case, if the ISTRIP flag is not set, a valid character of \377 is given to the application as \377, \377. If neither the IGNPAR nor the PARMRK flag is set, a framing or parity error (other than break) is given to the application as a single character \0.

INPCK
Enables input parity checking. If this flag is set, input parity checking is enabled. If not set, input parity checking is disabled. This allows for output parity generation without input parity errors.

ISTRIP
Strips characters. If this flag is set, valid input characters are first stripped to 7 bits. Otherwise, all 8 bits are processed.

INLCR
Maps a new-line character (NL) to a carriage return (CR) on input. If this flag is set, a received NL character is translated into a CR character.

IGNCR
Ignores CR character. If this flag is set, a received CR character is ignored and not read.

ICRNL
Maps a CR character to the NL character on input. If the ICRNL flag is set and the IGNCR flag is not set, a received CR character is translated into a NL character.

IUCLC
Maps uppercase to lowercase on input. If this flag is set, a received uppercase, alphabetic character is translated into the corresponding lowercase character.

IXON
Enables start and stop output control. If this flag is set, a received STOP character suspends output and a received START character restarts output. When the IXON flag is set, START and STOP characters are not read, but merely perform flow-control functions. When the IXON flag is not set, the START and STOP characters are read.

IXANY
Enables any character to restart output. If this flag is set, any input character restarts output that was suspended.

IXOFF
Enables start-and-stop input control. If this flag is set, the system transmits a STOP character when the input queue is nearly full and a START character when enough input has been read that the queue is nearly empty again.

IMAXBEL
Echoes the ASCII BEL character if the input stream overflows. Further input is not stored, but input already present in the input stream is not lost. If this flag is not set, no BEL character is echoed; the input in the input queue is discarded if the input stream overflows. This function also requires the IEXTEN bit to be set.

c_oflag
Specifies how the system treats output. The initial output-control value is “all bits clear.” The possible output modes are:

OPOST
Post-processes output. If this flag is set, output characters are post-processed as indicated by the remaining flags. Otherwise, characters are transmitted without change.
OLCUC
Maps lowercase to uppercase on output. If this flag is set, a lowercase alphabetic character is transmitted as the corresponding uppercase character. This flag is often used in conjunction with the IUCLC input mode.

ONLCR
Maps NL to CR-NL on output. If this flag is set, the NL character is transmitted as the CR-NL character pair.

OCRNL
Maps CR to NL on output. If this flag is set, the CR character is transmitted as the NL character.

ONOCR
Indicates no CR output at column 0. If this flag is set, no CR character is transmitted when at column 0 (first position).

ONLRET
NL performs CR function. If this flag is set, the NL character is assumed to do the carriage-return function. The column pointer is set to 0, and the delay specified for carriage return is used. If neither the ONLCR, OCRNL, ONOCR, nor ONLRET flag is set, the NL character is assumed to do the line-feed function only. The column pointer remains unchanged. The column pointer is set to 0 if the CR character is actually transmitted.

The delay bits specify how long a transmission stops to allow for mechanical or other movement when certain characters are sent to the terminal. The actual delays depend on line speed and system load.

OFILL
Uses fill characters for delay. If this flag is set, fill characters are transmitted for a delay instead of a timed delay. This is useful for high baud rate terminals that need only a minimal delay.

OFDEL
If this flag is set, the fill character is DEL. If this flag is not set, the fill character is NULL.

NLDLY
Selects the new-line character delays. This is the mask to use before comparing to NL0 and NL1:

NL0  Specifies no delay.
NL1  Specifies a delay of approximately 0.10 seconds. If the ONLRET flag is set, the carriage-return delays are used instead of the new-line delays. If the OFILL flag is set, two fill characters are transmitted.

CRDLY
Selects the carriage-return delays. This is the mask to use before comparing to CR0, CR1, CR2, and CR3:

CR0  Specifies no delay.
CR1  Specifies that the delay is dependent on the current column position. If the OFILL flag is set, this delay transmits two fill characters.
CR2  Specifies a delay of approximately 0.10 seconds. If the OFILL flag is set, this delay transmits four fill characters.
CR3  Specifies a delay of approximately 0.15 seconds.

TABDLY
Selects the horizontal-tab delays. This is the mask to use before comparing to TAB0, TAB1, TAB2, and TAB3. If the OFILL flag is set, any of these delays (except TAB3) transmit two fill characters.

TAB0  Specifies no delay.
TAB1  Specifies that the delay is dependent on the current column position. If the OFILL flag is set, two fill characters are transmitted.
TAB2  Specifies a delay of approximately 0.10 seconds.
TAB3  Specifies that tabs are to be expanded into spaces.

**BSDLY**
Selects the backspace delays. This is the mask to use before comparing to BS0 and BS1:
- **BS0**  Specifies no delay.
- **BS1**  Specifies a delay of approximately 0.05 seconds. If the **OFILL** flag is set, this delay transmits one fill character.

**VTDLY**
Selects the vertical-tab delays. This is the mask to use before comparing to VT0 and VT1:
- **VT0**  Specifies no delay.
- **VT1**  Specifies a delay of approximately 2 seconds.

**FFDLY**
Selects the form-feed delays. This is the mask to use before comparing to FF0 and FF1:
- **FF0**  Specifies no delay.
- **FF1**  Specifies a delay of approximately 2 seconds.

**c_cflag**
Describes the hardware control of the terminal. In addition to the basic control modes, this field uses the following control characters:

**CBAUD**
Specifies baud rate. These bits specify the baud rate for a connection. For any particular hardware, impossible speed changes are ignored.
- **B50**  50 baud.
- **B75**  75 baud.
- **B110**  110 baud.
- **B134**  134.5 baud.
- **B150**  150 baud.
- **B200**  200 baud.
- **B300**  300 baud.
- **B600**  600 baud.
- **B1200**  1200 baud.
- **B1800**  1800 baud.
- **B2400**  2400 baud.
- **B4800**  4800 baud.
- **B9600**  9600 baud.
- **B19200**  19200 baud.
- **B38400**  38400 baud.

**EXTA**  External A.
**EXTB**  External B.
CSIZE  Specifies the character size. These bits specify the character size, in bits, for both transmit and receive operations. The character size does not include the parity bit, if one is used:

- **CS5**  5 bits
- **CS6**  6 bits
- **CS7**  7 bits
- **CS8**  8 bits.

CSTOPB  Specifies number of stop bits. If this flag is set, 2 stop bits are sent; otherwise, only 1 stop bit is sent.

CREAD  Enables receiver. If this flag is set, the receiver is enabled. Otherwise, characters are not received.

PARENB  Enables parity. If this flag is set, parity generation and detection is enabled and a parity bit is added to each character.

PARODD  Specifies odd parity. If parity is enabled, the PARODD flag specifies odd parity if set. If parity is enabled and the PARODD flag is not set, even parity is used.

HUPCL  Hangs up on last close. If this flag is set, the line is disconnected when the last process closes the line or when the process terminates (when the `data terminal ready' signal drops).

CLOCAL  Specifies a local line. If this flag is set, the line is assumed to have a local, direct connection with no modem control. If not set, modem control (dial-up) is assumed.

CIBAUD  Specifies the input baud rate if different from the output rate.

PAREXT  Specifies extended parity for mark and space parity.

c_lflag  Controls various terminal functions. The initial value after an open is "all bits clear." In addition to the basic modes, this field uses the following mask name symbols:

- **ISIG**  Enables signals. If this flag is set, each input character is checked against the INTR, QUIT, SUSP, and DSUSP special control characters. If an input character matches one of these control characters, the function associated with that character is performed. If the ISIG flag is not set, checking is not done.

- **ICANON**  Enables canonical input. If this flag is set, it turns on canonical processing, which enables the erase and kill edit functions as well as the assembly of input characters into lines delimited by NL, EOF, and EOL characters. If the ICANON flag is not set, read requests are satisfied directly from the input queue. In this case, a read request is not satisfied until one of the following conditions is met:
  - The minimum number of characters specified by **MIN** are received.
  - The time-out value specified by **TIME** has expired since the last character was received. This allows bursts of input to be read, while still allowing single-character input.
The **MIN** and **TIME** values are stored in the positions for the EOF and EOL characters, respectively. The character values of **MIN** and **TIME** are converted to their ascii equivalents to get the numeric value. The time value represents tenths of seconds.

**XCASE**

Enables canonical uppercase and lowercase presentation. If this flag is set along with the **ICANON** flag, an uppercase letter (or the uppercase letter translated to lowercase by the **IUCLC** input mode) is accepted on input by preceding it with a `\` (backslash) character. The output is then also preceded by a backslash character. In this mode, the output generates and the input accepts the following escape sequences:

```
\ (grave)
  ``
\! (exclamation)
\~ (tilde)
\^ (caret)
\{ (left brace)
\} (right brace)
\ (backslash)
```

For example, A is input as `\a`, `\n` as `\n`, and `\N` as `\\n`.

**NOFLSH**

Disables queue flushing. If this flag is set, the normal flushing of the input and output queues associated with the INTR, QUIT, and SUSP characters is not done.

**FLUSHO**

Flushes the output. When this bit is set by typing the FLUSH character, data written to the terminal is discarded. A terminal can cancel the effect of typing the FLUSH character by clearing this bit.

**PENDIN**

Reprints pending input. If this flag is set, any input that is pending after a switch from raw to canonical mode is re-input the next time a read operation becomes pending or the next time input arrives. The **PENDIN** flag is an internal-state bit.

**IEXTEN**

Enables extended (implementation-defined) functions to be recognized from the input data. If this flag is not set, implementation-defined functions are not recognized, and the corresponding input characters are processed as described for the **ICANON**, **ISIG**, **IXON**, and **IXOFF** flags. Recognition of the following special control characters requires the **IEXTEN** flag to be set:

- **VEOL2**
- **VDSUSP**
- **VREPRINT**
- **VDISCRD**
- **VWERSE**
- **VLNEXT**

The functions associated with the following bits also require the **IEXTEN** flag to be set:

- **IMAXBEL**
- **ECHOKE**
- **ECHOPRT**
- **ECHOCTL**
TOSTOP
Sends a SIGTTOU signal when a process in a background process group tries to write to its controlling terminal. The SIGTTOU signal stops the members of the process group.

ECHO Enables echo. If this flag is set, characters are echoed as they are received.
When the ICANON is set, the following echo functions are also possible:

ECHO
Echoes the erase character as Backspace-Space-Backspace. If the ECHO and ECHOE flags are both set and the ECHOPRT flag is not set, the ERASE and WERASE characters are echoed as one or more ASCII Backspace-Space-Backspace sequences, which clears the last characters from the screen.

ECHOPRT
If the ECHO and ECHOPRT flags are both set, the first ERASE and WERASE character in a sequence are echoed as a \ (backslash), followed by the characters being erased. Subsequent ERASE and WERASE characters echo the characters being erased, in reverse order. The next non-erase character causes a / (slash) to be typed before the nonerase character is echoed. This function also requires the IEXTEN bit to be set.

ECHOKE
Backspace-Space-Backspace entire line on line kill. If this flag is set, the kill character is echoed by erasing the entire line from the screen (using the mechanism selected by the ECHOE and ECHOPRT flags). This function also requires the IEXTEN flag to be set.

ECHOK
Echoes the NL character after kill. If the ECHOK flag is set and the ECHOKE flag is not set, the NL character is echoed after the kill character is received. This emphasizes that the line is deleted.

ECHONL
Echoes the NL character. If the ECHONL flag is set, the NL character is echoed even if the ECHO flag is not set. This is useful for terminals that are set to "local echo" (also referred to as "half-duplex").

ECHOCTL
Echoes control characters (with codes between 0 and 37 octal) as ^X, where X is the character that results from adding 100 octal to the code of the control character. (For example, the character with octal code 1 is echoed as ^\). The ASCII DEL character (code 177 octal) is echoed as ^?. The ASCII TAB, NL, and START characters are not echoed. Unless escaped (preceded by a backslash), the EOF character is not echoed. As a result, because EOT is the default EOF character, terminals that respond to EOT are prevented from hanging up. This function also requires the IEXTEN flag to be set.

c_cc Specifies an array that defines the special control characters. The relative positions and initial values for each function are:

VINTR Indexes the INTR special character (Ctrl-c), which is recognized on input if the ISIG flag is set. The INTR character generates a SIGINT signal, which is sent to all processes in the foreground process group for which the terminal is the controlling terminal. If the ISIG flag is set, the INTR character is discarded when processed.

VQUIT Indexes the QUIT special character (Ctrl-\), which is recognized on input if the ISIG flag is set. The QUIT character generates a SIGQUIT signal, which is sent to all processes in the foreground process group for which the terminal is the controlling terminal, and writes a core image file into the current working directory. If the ISIG flag is set, the QUIT character is discarded when processed.

VERASE Indexes the ERASE special character (Backspace), which is recognized on input if the
ICANON flag is set. The ERASE character does not erase beyond the beginning of the line as delimited by a NL, EOL, EOF, or EOL2 character. If the ICANON flag is set, the ERASE character is discarded when processed.

VKILL Indexes the KILL special character (Ctrl-u), which is recognized on input if the ICANON flag is set. The KILL character deletes the entire line, as delimited by a NL, EOL, EOF, or EOL2 character. If the ICANON flag is set, the KILL character is discarded when processed.

VEOF Indexes the EOF special character (Ctrl-d), which is recognized on input if the ICANON flag is set. When EOF is received, all the characters waiting to be read are immediately passed to the process, without waiting for a new line, and the EOF is discarded. If the EOF is received at the beginning of a line (no characters are waiting), a character count of zero is returned from the read, indicating an end-of-file. If the ICANON flag is set, the EOF character is discarded when processed.

VEOL Indexes the EOL special character (Ctrl-@ or ASCII NULL), which is recognized on input if the ICANON flag is set. EOL is an additional line delimiter, like NL, and is not normally used.

VEOL2 Indexes the EOL2 special character (Ctrl-@ or ASCII NULL), which is recognized on input if the ICANON and IEXTEN flags are set. EOL2 is an additional line delimiter, like NL, and is not normally used.

VSTART Indexes the START special character (Ctrl-q), which is recognized on input if the IXON flag is set, and generated on output if the IXOFF flag is set. The START character can be used to resume output that has been suspended by a STOP character. If the IXON flag is set, the START character is discarded when processed. While output is not suspended, START characters are ignored and not read. VSTRT is an alias for VSTART.

VSTOP Indexes the STOP special character (Ctrl-s), which is recognized on input if the IXON flag is set, and generated on output if the IXOFF flag is set. The STOP character can be used to with terminals to prevent output from disappearing before it can be read. If the IXON flag is set, the STOP character is discarded when processed. While output is suspended, STOP characters are ignored and not read.

VSUSP Indexes the SUSP special character (Ctrl-z), which is recognized on input if the ISIG flag is set. The SUSP character generates a SIGTSTP signal, which is sent to all processes in the foreground process group for which the terminal is the controlling terminal. If the ISIG flag is set, the SUSP character is discarded when processed.

VDSUSP Indexes the DSUSP special character (Ctrl-y), which is recognized on input if the ISIG and IEXTEN flags are set. The DSUSP character generates a SIGTSTP signal as the SUSP character does, but the signal is sent when a process in the foreground process group attempts to read the DSUSP character, rather than when DSUSP is typed. If the ISIG and IEXTEN flags are set, the DSUSP character is discarded when processed.

VREPRINT Indexes the REPRINT special character (Ctrl-r), which is recognized on input if the ICANON and IEXTEN flags are set. The REPRINT character reprints all characters, preceded by a new line, that have not been read. If the ICANON and IEXTEN flags are set, the REPRINT character is discarded when processed.

VDISCRD Indexes the DISCARD special character (Ctrl-o), which is recognized on input if the ICANON and IEXTEN flags are set. The DISCARD character causes subsequent output to
be discarded until another DISCARD character is typed, more input arrives, or the condition is cleared by a program. If the ICANON and IEXTEN flags are set, the DISCARD character is discarded when processed.

**VWERSE**
Indexes the WERASE special character (Ctrl-w), which is recognized on input if the ICANON and IEXTEN flags are set. The WERASE character causes the preceding word to be erased. The WERASE character does not erase beyond the beginning of the line as delimited by a NL, EOL, EOF, or EOL2 character. If the ICANON and IEXTEN flags are set, the WERASE character is discarded when processed.

**VLNEXT**
Indexes the LNEXT (literal next) special character (Ctrl-v), which is recognized on input if the ICANON and IEXTEN flags are set. The LNEXT character causes the special meaning of the next character to be ignored so that characters can be input without being interpreted by the system. If the ICANON, ECHO, and IEXTEN flags are set, the LNEXT character is replaced by a ^-Backspace sequence when processed.

**VMIN**
Indexes the MIN value, which is not a special character. The use of the MIN value is described in the discussion of noncanonical mode input processing in “Idterm Line Discipline” in AIX 5L Version 5.3 General Programming Concepts: Writing and Debugging Programs.

**VTIME**
Indexes the TIME value, which is not a special character. The use of the TIME value is described in the discussion of noncanonical mode input processing in “Idterm Line Discipline” in AIX 5L Version 5.3 General Programming Concepts: Writing and Debugging Programs.

The character values for the following control characters can be changed:

<table>
<thead>
<tr>
<th>Control Character</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTR</td>
<td>EOF</td>
</tr>
<tr>
<td>QUIT</td>
<td>EOL</td>
</tr>
<tr>
<td>ERASE</td>
<td>EOL2</td>
</tr>
<tr>
<td>KILL</td>
<td>START</td>
</tr>
<tr>
<td></td>
<td>STOP</td>
</tr>
<tr>
<td></td>
<td>SUSP</td>
</tr>
<tr>
<td></td>
<td>DSUSP</td>
</tr>
<tr>
<td></td>
<td>REPRINT</td>
</tr>
<tr>
<td></td>
<td>DISCARD</td>
</tr>
<tr>
<td></td>
<td>WERASE</td>
</tr>
<tr>
<td></td>
<td>LNEXT</td>
</tr>
</tbody>
</table>

The ERASE, KILL, and EOF characters can also be escaped (preceded by a backslash) so that no special processing is done.

**Parameter Value Definitions**

The following values for the OptionalActions parameter of the tcsetattr subroutine are also defined in the termios.h file:

<table>
<thead>
<tr>
<th>Parameter Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCSANOW</td>
<td>Immediately sets the parameters associated with the terminal from the referenced termios structure.</td>
</tr>
<tr>
<td>TCSADRAIN</td>
<td>Waits until all output written to the object file has been transmitted before setting the terminal parameters from the termios structure.</td>
</tr>
<tr>
<td>TCSAFLUSH</td>
<td>Waits until all output written to the object file has been transmitted and until all input received but not read has been discarded before setting the terminal parameters from the termios structure.</td>
</tr>
</tbody>
</table>

The following values for the QueueSelector parameter of the tcflush subroutine are also defined in this header file:

<table>
<thead>
<tr>
<th>Parameter Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCIFLUSH</td>
<td>Flushes data that is received but not read.</td>
</tr>
<tr>
<td>TCOFLUSH</td>
<td>Flushes data that is written but not transmitted.</td>
</tr>
<tr>
<td>TCOFLUSH</td>
<td>Flushes data that is received but not read as well as data that is written but not transmitted.</td>
</tr>
</tbody>
</table>
The following values for the Action parameter of the tcflow subroutine are also defined in the termios.h file:

- **TCOOFF** Suspend the output of data by the object file named in the tcflow subroutine.
- **TCOON** Restarts data output that was suspended by the TCOOFF action.
- **TCIOFF** Transmits a stop character to stop data transmission by the terminal device.
- **TCION** Transmits a start character to start or restart data transmission by the terminal device.

Modem Control Operations

The following ioctl operations, used for modem control, are an extension to the POSIX line discipline interface. To use these operations in a program, the program must contain an #include statement for the ioctl.h file.

- **TIOCMBIS** The argument to this command is a pointer to an integer that turns on the control lines specified by the integer mask value. No other control lines are affected.
- **TIOCMBIC** The argument to this command is a pointer to an integer that turns off the control lines specified by the integer mask value. No other control lines are affected.
- **TIOCMGET** Gets all modem bits. The argument to this command is a pointer to an integer where the current state of the modem status lines is stored. Which modem status and modem control lines are supported depends on the capabilities of the hardware and the hardware's device driver.
- **TIOCMSET** Sets all modem bits. The argument to this command is a pointer to an integer containing a new set of modem bits. The modem control bits use these bits to turn the modem control lines on or off, depending on whether the bit for that line is set or clear. Any modem status bits are ignored. The actual modem control lines which are supported depend on the capabilities of the hardware and the hardware's device driver.

The integer specifies one of the following modem control or status lines on which the modem control ioctl command operates:

- **TIOCМ_LE** Line enable
- **TIOCМ_DTR** Data terminal ready
- **TIOCМ_RTS** Request to send
- **TIOCМ_ST** Secondary transmit
- **TIOCМ_SR** Secondary receive
- **TIOCМ_CTS** Clear to send
- **TIOCМ_CAR** Carrier detect
- **TIOCМ_CD** TIOCМ_CAR
- **TIOCМ_RNG** Ring
- **TIOCМ_RI** TIOCМ_RNG
- **TIOCМ_DSR** Data set ready.
TIOCMIWAIT  Wait for modem status line to change status.

The argument is a pointer to an integer mask value specifying the modem status line(s) on which to wait for a status change, and can consist of one or more of the following values:

- **TIOCМ_CTS**
  - Clear to send
- **TIOCМ_CAR**
  - Carrier detect
- **TIOCМ_CD**
  - TIOCМ_CAR
- **TIOCМ_RNG**
  - Ring
- **TIOCМ_RI**
  - TIOCМ_RNG
- **TIOCМ_DSR**
  - Data set ready.

The request blocks until one of the specified lines changes status, then returns to the caller. Note that this ioctl blocks even if O_NDELEY or O_NONBLOCK is set.

If none of the specified lines changes status, the ioctl can block indefinitely, so it should be used in conjunction with an alarm() timer.

If TIOCМ_RNG or TIOCМ_RI is specified, the transition is reported only when the status line transitions from on to off due to hardware restrictions.

**Note:** Correct operation of this ioctl depends on correct cabling.

---

**Related Information**

The **termiox.h** file, **types.h** file.

The **csh** command, **getty** command, **ksh** command, **stty** command, **tset** command.

The **cfgetispeed**, **cfgetospeed**, **cfsetispeed**, **cfsetospeed** subroutine, **ioctl** subroutine, **sigvec** subroutine, **tcdrain** subroutine, **tcflow** subroutine, **tcflush** subroutine, **tcgetattr** subroutine, **tcsendbreak** subroutine, **tcsetattr** subroutine.

[tty Subsystem Overview in AIX 5L Version 5.3 General Programming Concepts: Writing and Debugging Programs.]

---

**termiox.h File**

**Purpose**
Defines the structure of the **termiox** file, which provides the extended terminal interface.

**Description**
The **termiox.h** file contains an extended terminal interface to support asynchronous hardware flow control. It defines the **termiox** structure and ioctl operations using this structure. The **termiox** structure in the **termiox.h** file contains the following fields:

- x_hflag
- x_cflag
- x_rflag
The `termiox.h` file also supports ioctl hardware flow control operations.

**Field**  
**Descriptions**

- **x_hflag**  
  Describes the hardware flow control mode. The possible modes are:

  - **CDXON**  
    Enables CD hardware flow control on output. When set, output will occur only if the `receive line signal detector` (CD) line is raised by the connected device. If the CD line is dropped by the connected device, output is suspended until the CD line is raised.

  - **CTSXON**  
    Enables CTS hardware flow control on output. When set, output will occur only if the `clear to send` (CTS) line is raised by the connected device. If the CTS line is dropped by the connected device, output is suspended until the CTS line is raised.

  - **DTRXOFF**  
    Enables DTR hardware flow control on input. When set, the `data terminal ready` (DTR) line is raised. If the port needs to have its input stopped, it will drop the DTR line. It is assumed that the connected device will stop its output until DTR is raised.

  - **RTXOFF**  
    Enables RTS hardware flow control on input. When set, the `request to send` (RTS) line is raised. If the port needs to have its input stopped, it will drop the RTS line. It is assumed that the connected device will stop its output until RTS is raised.

It is not possible to use simultaneously the following flow control modes:

- RTS and DTR
- CTS and CD.

Different hardware flow control modes may be selected by setting the appropriate flags. For example:

- Bi-directional RTS/CTS flow control by setting **RTXOFF** and **CTSXON**
- Bi-directional DTR/CTS flow control by setting **DTRXOFF** and **CTSXON**
- Modem control or uni-directional CTS flow control by setting **CTSXON**.

- **x_cflag**  
  Reserved for future use.

- **x_rflag**  
  Reserved for future use.

- **x_sflag**  
  Describes the open discipline. This field must be set before the first open; it is usually done at configuration time. The possible disciplines are:

  - **DTR_OPEN**  
    DTR open discipline. On open, the discipline raises the `data terminal ready` (DTR) and `request to send` (RTS) lines, and waits for the `data carrier detect` (DCD) line to be raised. If the port is opened with the **O_NDELAY** or **O_NONBLOCK** flags, the wait is not done. The DTR and RTS lines are dropped at close time.

  - **WT_OPEN**  
    World trade open discipline. On open, the discipline behaves like the DTR open discipline if not in CDSTL mode. In CDSTL mode, the discipline does not raise the DTR line until the `ring indicate` (RI) line is raised. The DTR line is dropped when the DSR line drops for more than 20 milliseconds.

### Hardware Flow Control Operations

The following ioctl operations are used for hardware flow control. To use these operations in a program, the program must contain an `#include` statement for the `ioctl.h` file. The argument to these operations is a pointer to a `termiox` structure.

- **TCGETX**  
  Gets the terminal parameters. The current terminal parameters are stored in the structure.

- **TCSETX**  
  Sets the terminal parameters immediately. The current terminal parameters are set according to the structure. The change is immediate.
TCSETXW
Sets the terminal parameters after end of output. The current terminal parameters are set according to
the structure. The change occurs after all characters queued for output have been transmitted. This
operation should be used when changing parameters will affect output.

TCSETXF
Sets the terminal parameters after end of output and flushes input. The current terminal parameters are
set according to the structure. All characters queued for output are first transmitted, then all characters
queued for input are discarded, and then the change occurs.

Related Information
The `termios.h` file.
The `ioctl` subroutine.

Related System Overview in AIX 5L Version 5.3 General Programming Concepts: Writing and Debugging
Programs.

`types.h` File

Purpose
Defines primitive system data types.

Description
The `/usr/include/sys/types.h` file defines data types used in system source code. Since some system
data types are accessible to user code, they can be used to enhance portability across different machines
and operating systems. For example, the `pid_t` type allows for more processes than the unsigned short
`ushort_t` type, and the `dev_t` type can be 16 bits rather than 32 bits.

Standard Type Definitions
The `types.h` file includes the following standard type definitions, which are defined with a `typedef`
statement:

- `daddr_t`: Used for disk addresses, except in i-nodes on disk. The `/usr/include/sys/filsys.h` file format
describes the format of disk addresses used in i-nodes.
- `caddr_t`: Core (memory) address.
- `clock_t`: Used for system times as specified in `CLK_TCKs`.
- `ino_t`: File system i-node number.
- `cnt_t`: File system reference count type.
- `dev_t`: Major and minor parts of a device code specify the kind of device and unit number of the device and
depend on how the system is customized.
- `chan_t`: Channel number (the minor's minor).
- `off_t`: File offset, measured in bytes from the beginning of a file or device. `off_t` is normally defined as a
  signed, 32-bit integer. In the programming environment which enables large files, `off_t` is defined to
  be a signed, 64-bit integer.
- `offset_t`: 64-bit file offset, measured in bytes from the beginning of a file or device.
- `off64_t`: 64-bit file offset, measured in bytes from the beginning of a file or device.
- `soff_t`: 32-bit file offset, measured in bytes from the beginning of a file or device.
- `paddr_t`: Real address.
- `key_t`: IPC key.
- `time_t`: Timer ID. Times are encoded in seconds, since 00:00:00 UCT, January 1, 1970.
- `nlink_t`: Number of file links.
- `mode_t`: File mode.
- `uid_t`: User ID.
- `gid_t`: Group ID.
- `mid_t`: Module ID.
pid_t     Process ID.
slab_t    Security label.
mtyp_t    Interprocess communication (IPC) message type.
size_t    Data type is used for sizes of objects.
ssize_t   Data type is used for a count of bytes or an error indication.
uchar_t   Unsigned char.
ushort_t  Unsigned short.
uint_t    Unsigned int.
ulong_t   Unsigned long.

Unsigned Integers and Addresses
The types.h file also includes the following type definitions for unsigned integers and addresses:

typedef struct _quad { long val[2]; } quad;
typedef long    swblk_t;
typedef unsigned long size_t;

The following type definitions are for BSD compatibility only:

typedef unsigned char   u_char;
typedef unsigned short  u_short;
typedef unsigned int    u_int;
typedef unsigned long   u_long;

Related Information
The values.h file.
The filsys.h file format.

unistd.h File

Purpose
Defines implementation characteristics identified by POSIX standard.

Description
The /usr/include/unistd.h file includes files that contain definitions that are required for compatibility with the POSIX standard:

access.h     Defines symbolic constants for the access subroutine.

The unistd.h file also defines symbolic constants for the pathconf, fpathconf, and sysconf subroutines. The unistd.h file also defines the following symbols, which are used by POSIX applications to determine implementation characteristics:

_POSIX_JOB_CONTROL    POSIX-compatible job control is supported.
_POSIX_SAVED_IDS      An exec subroutine saves the effective user and group IDs.
_POSIX_VERSION        The version of the POSIX standard with which this version of the operating system complies. The value of this symbol is 198808L.
_POSIX_CHOWN_RESTRICTED The use of the chown function is restricted to a process with the appropriate privileges. The group ID of a file can be changed only to the effective group ID or a supplementary group ID of the process. The value of this symbol is -1.
_POSIX_VDISABLE       The terminal special characters, which are defined in the termios.h file, can be disabled if this character value is defined by the tcsetattr subroutine. The value of this symbol is -1.
Path name components that are longer than `NAME_MAX` will generate an error.

The `unistd.h` file also defines the following symbol, which is used by X/OPEN applications:

```
_XOPEN_VERSION
```

The version of the X/OPEN standard with which this version of the operating system complies.

**Related Information**

The `limits.h` file, `sys/types.h` file, `termios.h` file, `values.h` file.

The `access` subroutine, `exec` subroutine.

---

**utmp.h File**

**Purpose**

Defines the structures of certain user and accounting information files.

**Description**

The structure of the records in the `utmp`, `wtmp`, and `failedlogin` files is defined in the `/usr/include/utmp.h` file. The `utmp` structure in this header file contains the following fields:

- `ut_user` User login name.
- `ut_line` Device name (console or lnxx). The maximum length of a string in this field is 11 characters plus a null character. When accounting for something other than a process, the following special strings or formats are allowed:
  - `RUNLVL_MSG` Run level: specifically, the run level of the process.
  - `BOOT_MSG` System boot: specifically, the time of the initial program load (IPL).
  - `OTIME_MSG` Old time: specifically, the time of login.
  - `NTIME_MSG` New time: specifically, the time idle.
- `ut_pid` Process ID.
ut_type  Type of entry, which can be one of the following values:

EMPTY
Unused space in file.

RUN_LVL
The run level of the process, as defined in the `inittab` file.

BOOT_TIME
The time at which the system was started.

OLD_TIME
The time at which a user logged on to the system.

NEW_TIME
The amount of time the user is idle.

INIT_PROCESS
A process spawned by the `init` command.

LOGIN_PROCESS
A `getty` process waiting for a login.

USER_PROCESS
A user process.

DEAD_PROCESS
A zombie process.

ACCOUNTING
A system accounting process.

UTMAXTYPE ACCOUNTING
The largest legal value allowed in the `ut_type` field.

Embedded within the `utmp` structure is the `exit_status` structure, which contains the following fields:

- `e_termination`: Termination status of a process.
- `e_exit`: Exit status of a process, marked as the `DEAD_PROCESS` value.
- `ut_time`: Time at which the entry was made.

**Examples**

```c
#ifndef -H-UTMP
#define _H_UTMP
#define UTMP_FILE  "/etc/utmp"
#define WTMP_FILE  "/var/adm/wtmp"
#define ILOG_FILE  "/etc/.ilog"
#define ut_name   ut_user

struct utmp
{
    char  ut_user[256];  /* User login name */
    char  ut_id[14];    /* /etc/inittab id */
    char  ut_line[64];  /* device name (console, lnxx) */
    pid_t ut_pid;       /* process id */
    short ut_type;      /* type of entry */
#ifdef __64BIT__
    int __time_t_space; /* for 32vs64-bit time_t PPC */
#endif
    time_t ut_time;     /* time entry was made */

    struct exit_status
    {
        short e_termination; /* Process termination status */
        short e_exit;        /* Process exit status */
    }
};
```
ut_exit;                   /* The exit status of a process
    */
char ut_host[256];       /* host name */
int __dbl_word_pad;     /* for double word alignment */
int __reservedA[2];
int __reservedV[6];
};
/* Definitions for ut_type */
#define EMPTY 0
#define RUN_LVL 1
#define BOOT_TIME 2
#define OLD_TIME 3
#define NEW_TIME 4
#define INIT_PROCESS 5 /* Process spawned by "init" */
#define LOGIN_PROCESS 6 /* A "getty" process */
#define USER_PROCESS 7 /* waiting for login */
#define DEAD_PROCESS 8 /* A user process */
#define ACCOUNTING 9
#define UTMAXTYPE ACCOUNTING /* Largest legal value */
    /* of ut_type */

/* Special strings or formats used in the */
/* "ut_line" field when accounting for */
/* something other than a process. */
/* No string for the ut line field can be more */
/* than 11 chars + a NULL in length. */
#define RUN_LVL_MSG "run-level %c"
#define BOOT_MSG "system boot"
#define OTIME_MSG "old time"
#define TIME_MSG "new time"
#endif
/* _H_UTMP */

Note: The who command extracts information from the /etc/utmp, /var/adm/wtmp, and
/etc/security/failedlogin files.

Files
/etc/utmp
The path to the utmp file, which contains a record of users logged
in to the system.
/var/adm/wtmp
The path to the wtmp file, which contains accounting information
about logged-in users.
/etc/security/failedlogin
The path to the failedlogin file, which contains a list of invalid login
attempts.

Related Information
The getty command, init command, login command, who command, write command.

The utmp, wtmp, failedlogin file format.

values.h File

Purpose
Defines machine-dependent values.
Description
The /usr/include/values.h file contains a set of constants that are conditionally defined for particular processor architectures. The model for integers is assumed to be a ones or twos complement binary representation, in which the sign is represented by the value of the high-order bit.

BITS(type) Number of bits in the specified data type
HIBITS Short integer with only the high-order bit set (0x8000)
HIBITL Long integer with only the high-order bit set (0x80000000)
HIBITI Regular integer with only the high-order bit set (same as the HIBITL value)
MAXSHORT Maximum value of a signed short integer (0x7FFF = 32,767)
MAXLONG Maximum value of a signed long integer (0x7FFFFFFF = 2,147,483,647)
MAXINT Maximum value of a signed regular integer (same as the MAXLONG value)
MAXFLOAT Maximum value of a single-precision floating-point number
MAXDOUBLE Maximum value of a double-precision floating-point number
LN_MAXDOUBLE Natural logarithm of the MAXDOUBLE value
MINFLOAT Minimum positive value of a single-precision floating-point number
MINDouble Minimum positive value of a double-precision floating-point number
FSIGNIF Number of significant bits in the mantissa of a single-precision floating-point number
DSIGNIF Number of significant bits in the mantissa of a double-precision floating-point number
FMAXEXP Maximum exponent of a single-precision floating-point number
DMAXEXP Maximum exponent of a double-precision floating-point number
FMINEXP Minimum exponent of a single-precision floating-point number
DMINEXP Minimum exponent of a double-precision floating-point number
FMAXPOWTWO Largest power of two that can be exactly represented as a single-precision floating-point number
DMAXPOWTWO Largest power of two that can be exactly represented as a double-precision floating-point number

Related Information
The math.h file, types.h file.

vmount.h File
Purpose
Defines the structure of the data associated with a virtual file system.

Description
The /usr/include/sys/vmount.h file defines the vmount structure. Each active virtual file system (VFS) has a vmount structure associated with it. The vmount structure contains the mount parameters (such as the mount object and the mounted-over object) for that VFS. The vmount data is created when the VFS is mounted. The mntctl subroutine returns the VFS data.

The vmount structure contains the following fields to describe fixed-length data:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vmt_revision</td>
<td>The revision code in effect when the program that created this VFS was compiled.</td>
</tr>
<tr>
<td>vmt_length</td>
<td>The total length of the structure and data. This will always be a multiple of the word size (4 bytes).</td>
</tr>
<tr>
<td>vmt_fsid</td>
<td>The two-word file system identifier; the interpretation of this identifier depends on the vmt_gstype field.</td>
</tr>
<tr>
<td>vmt_vfsnumber</td>
<td>The unique identifier of the VFS. Virtual file systems and their identifiers are deleted at IPL (initial program load).</td>
</tr>
<tr>
<td>vmt_time</td>
<td>The time at which the VFS was created.</td>
</tr>
<tr>
<td>vmt_flags</td>
<td>The general mount flags, for example: READONLY, REMOVABLE, DEVICE, REMOTE.</td>
</tr>
</tbody>
</table>
The type of the general file system. Possible values are:

- **MNT_JFS**
  Journaled file system (JFS)

- **MNT_NFS**
  SUN network file system

- **MNT_CDROM**
  CD-ROM file system

The remaining fields in the `vmount` structure describe variable-length data. Each entry in the `vmt_data` array specifies the offset from the start of the `vmount` structure at which a data item appears, as well as the length of the data item.

- `vmt_off`: Offset of the data, aligned on a word (32-bit) boundary.
- `vmt_size`: Actual size of the data in bytes.
- `vmt_data[VMT_OBJECT]`: Name of the device, directory, or file that is mounted.
- `vmt_data[VMT_STUB]`: Name of the device, directory, or file that is mounted over.
- `vmt_data[VMT_HOST]`: Short (binary) name of the host that owns the mounted object.
- `vmt_data[VMT_HOSTNAME]`: Long (character) name of the host that owns the mounted object.
- `vmt_data[VMT_INFO]`: Binary information passed to the file system implementation that supports this object; the contents of this field are specific to the generic file system (GFS) type defined by the `vmt_gfstype` field.
- `vmt_data[VMT_ARGS]`: Character-string representation of the arguments supplied when the VFS was created.

**Related Information**

The `mntctl` subroutine, `umount` or `uvmount` subroutine, `vmount or mount` subroutine.

---

**wctype.h File**

**Purpose**

Contains wide-character classification and mapping utilities.

**Syntax**

```c
#include <wctype.h>
```

**Description**

The `wctype.h` header defines the following data types through typedef:

- `wint_t`: As described in `wchar.h`.
- `wctrans_t`: A scalar type that can hold values that represent locale-specific character mappings.
- `wctype_t`: As described in `wchar.h`.

The `wctype.h` header declares the following as functions and may also define them as macros. Function prototypes must be provided for use with an ISO C compiler.

```c
int iswalnum(wint_t);
int iswalpha(wint_t);
int iswcntrl(wint_t);
int iswdigit(wint_t);
int iswgraph(wint_t);
int iswlower(wint_t);
int iswprint(wint_t);
```
The **wctype.h** defines the following macro name:

**WEOF** Constant expression of type `wint_t` that is returned by several MSE functions to indicate end-of-file.

For all functions described in this header that accept an argument of type `wint_t`, the value will be representable as a `wchar_t` or will equal the value of **WEOF**. If this argument has any other value, the behaviour is undefined.

The behaviour of these functions is affected by the LC_CTYPE category of the current locale.

Inclusion of the **wctype.h** header may make visible all symbols from the headers **ctype.h**, **stdio.h**, **stdarg.h**, **stdlib.h**, **string.h**, **stddef.h**, **time.h** and **wchar.h**.

**Related Information**

The [iswalnum, iswalpha, iswcntrl, iswdigit, iswgraph, iswlower, iswprint, iswpunct, iswspace], [iswupper, iswxdigit, iswctype, setlocale], [towctrans, towlower, towupper, wctrans], and **wctype** subroutines.

The **locale.h** and **wchar.h** header files.

**wlm.h File**

**Purpose**

Defines the constants, data structures and function prototypes used by the Workload Manager (WLM) Application Programming Interface (API) subroutines.

**Description**

The **wlm.h** file defines the **wlm_args**, **wlm_assign**, **wlm_info**, **wlm_bio_class_info_t**, and **wlm_bio_dev_info_t** and **wlm_proc_info** structures. These structures are used by the WLM API functions in the **libwlm.a** library.

**The wlm_args Structure**

The **wlm_args** structure is used to pass class information to WLM when using the API functions to create, modify or delete a class. The **wlm_args** structure contains the following fields:

**versflags**

Specifies the 4 high-order bits that contain a version number used by the API to maintain binary compatibility if the data structures are ever modified. The rest of the integer is used to pass flags to the subroutines when needed.

This field should be initialized with a logical OR operation between the version number **WLM_VERSION** and whatever flags are needed by the target subroutine. One flag common to all the API calls is **WLM_MUTE**, which is used to suppress the output of error messages from the WLM library to **STDERR**.
confdir
Specifies a null-terminated string. This field must be initialized with the name of the WLM configuration to which the target subroutine applies (when applicable, depending on the particular one).

Alternatively, this field can be set to a null string (\0). The null string indicates that the class addition or modification is to be applied only to the WLM kernel data, not to the class description files.

class
This field is a structure of type struct class_definition that contains all the information pertaining to the superclass or subclass that is needed by the target subroutine. The fields in this structure can be initialized by a call to the wlm_init_class_definition subroutine so that you only need to initialize the fields you wish to modify.

The main structure in the wlm_init_class_definition subroutine is the class description, struct class_descr, with the following fields:

res
Specifies an array of type struct wlm_bounds that contains the following fields for each resource type and for each total limit:

min
Specifies the minimum value, which is between 0 (the default) and 100 (unused for total limits).

shares
Specifies the shares number, which is a value between 1 and 65535. The value -1 (default) indicates that the given resource is not managed by WLM for the class (unused for total limits).

wlmu
The union which contains the softmax (for all resources but total limits) and unit (for total limits only) fields:

softmax
Specifies the soft maximum limit, which is a value between 0 and 100 (default). The value must be greater than or equal to the value of the min field.

unit
A string (3 characters maximum) that specifies the units that apply to the hardmax value for total limits. To let the units remain undefined, set the softmax field to WLM_UNIT_UNDEF. For WLM_RES_TOTALCONNECT and WLM_RES_TOTALCPU, the default unit is "s" (seconds) and other values are "m" (minutes), "h" (hours), "d" (days), and "w" (weeks). For WLM_RES_TOTALDISKIO, the default unit is "KB" (kilobytes) and other values are "MB" (megabytes), "GB" (gigabytes), "TB" (terabytes), "PB" (petabytes), "EB" (exabytes). The other total limits do not have units. For WLM_RES_TOTALVMEM and WLM_RES_PROCMEM, set the unit to "MB", "GB", and "TB".

hardmax
For all resources but total limits. Specifies the hard maximum limit, which is a value between 0 and 100 (default). The value must be greater than or equal to each of the values of the min and softmax fields. For total limits, this parameter specifies their value, possibly along with the unit field. If the user does not specify a units value, the default units value is used. However, the user can specify a units value other than the default. The default (total limit unspecified) is WLM_HARDMAX_UNDEF.

The resource types are defined as WLM_RES_CPU, WLM_RES_MEM, WLM_RES_BIO, and total limits are defined as WLM_RES_TOTALCPU (total CPU time for a process), WLM_RES_TOTALDISKIO (total disk IOs for a process), WLM_RES_TOTALCONNECT (total Connection time), WLM_RES_TOTALPROC (total number of processes), WLM_RES_TOTALTHRD (total number of Threads), WLM_RES_TOTALLOGIN (total
number of login sessions), \texttt{WLM\_RES\_TOTALVMEM} (total virtual memory usage for a class) and \texttt{WLM\_RES\_PROCVMEM} (total virtual memory usage for a process). Each value represents the index in the array of the element, corresponding to the type of resource or total limit.

\textbf{tier} Specifies the tier number for the class, which is a value between 0 (default) and 9

\textbf{inheritance} Specifies how a new process is classified. A value of 0 (the default) indicates that a new process should be classified using the class assignment rules when calling the \texttt{exec} subroutine. A value of 1 indicates that the process inherits the class assignment from its parent process.

\textbf{localshm} Indicates whether memory segments in this class remain local to the class (value 1) or if they go to the Shared class (value 0, the default), when accessed by a process belonging to another class.

\textbf{assign\_uid} Specifies the user ID of the user allowed to manually assign processes to the class. The value must be a valid user ID.

The default when this attribute is not specified is that no user is authorized (WLM\_NOGUID).

\textbf{assign\_gid} Specifies the group ID of the users allowed to manually assign processes to the class.

The value must be a valid group ID. The value must be a valid group ID. The default when this attribute is not specified is that no group is authorized (WLM\_NOGUID).

If both the \texttt{assign\_uid} and \texttt{assign\_gid} fields are the default value, only the root user can assign processes to the class.

\textbf{admin\_uid} Specifies the user ID of the user allowed to administer the subclasses of the superclass (this attribute is valid only for superclasses)

\textbf{admin\_gid} Specifies the group ID of the users allowed to administer the subclasses of the superclass (this attribute is valid only for superclasses)

If both the \texttt{admin\_uid} and \texttt{admin\_gid} fields are left to their default value (WLM\_NOGUID), only the root user can administer the subclasses of this superclass.

\textbf{name} Specifies the null-terminated full name of the class. The value must be in the format \texttt{super\_name} for a superclass and \texttt{super\_name.sub\_name} for a subclass. The superclass name and subclass name are limited to 16 characters each. This field has no default value.

In addition to the class description fields, the \texttt{class\_definition} structure adds two more fields:

\textbf{rset\_name} Specifies a null-terminated character string containing the name of the resource set (partition) that the class is restricted to, when applicable. The default is that the class can access all the resources on the system.

\textbf{descr\_field} Specifies a null-terminated character string containing the description text of the class.

This field is optional and has no default.

\textbf{delshm} Specifies whether to delete shared memory segments if the last process referencing the segment was killed due to a virtual memory limit. Valid values are “yes” and “no” (default).
vmenforce

Specifies whether the faulting process (value "proc", default) or all processes in a class (value "class") are killed when a virtual memory limit is reached.

The wlm_assign Structure

The wlm_assign structure is used to manually assign processes or groups of processes to a specified superclass or subclass using the wlm_assign subroutine. The wlm_assign structure contains the following fields:

- **wa_versflags**: Specifies the 4 high-order bits containing a version number. This version number is used by the API to maintain binary compatibility if the data structures are ever modified. The rest of the integer is used to pass flags to the subroutines when needed.

  This field should be initialized with the version number WLM_VERSION. The flag WLM_MUTE can be used to suppress the output of error messages from the WLM library on stderr.

- **wa_pids**: Specifies the address of an array containing the process identifiers (PIDs) of the processes to be manually assigned.

- **wa_pid_count**: Specifies the number of PIDs in the array above.

- **wa_pgids**: Specifies the address of an array containing the process group IDs (PGIDs) of the process groups to be manually assigned.

- **wa_pgid_count**: Specifies the number of PGIDs in the array above.

- **wa_classname**: Specifies the full name of the superclass or the subclass of the class to which you want to manually assign processes.

The wlm_info Structure

The wlm_info structure is used to extract information about the current configuration parameters and current resource utilization of the active classes using the wlm_get_info subroutine. The wlm_info structure contains the following fields:

- **i_descr**: Specifies the class description of type struct class_descr.

- **i_regul**: Specifies the per-resource-type array of structures, which are of the type struct wlm_regul, containing the following fields:

  - **consum**: Specifies the resource consumption of the class. This value is expressed as a percentage of the total resource available.

  - **total**: Specifies the 64-bit number that represents the total amount of the resource consumed by the class since its creation (or since WLM was started). The value can be the number of milliseconds for CPU or the total number of 512-byte blocks for disk I/O. This field is left null (not significant) for memory.

The indexes into the array of the various resources are defined as WLM_RES_CPU, WLM_RES_MEM and WLM_RES_BIO.

- **i_class_id**: Specifies the class identifier (index of the class in the kernel class_control_block (ccb) table).

- **i_cl_pri**: Specifies the priority delta applied to the threads in the class for CPU regulation.

- **i_cl_inuse**: Specifies the current number of processes in the class.

- **i_cl_nblogin**: Specifies the current number of logins in the class.

- **i_cl_nthreads**: Specifies the current number of threads in the class.

- **i_cl_npages**: Specifies the number of memory pages currently allocated to the class.

- **i_cl_nvpages**: Specifies the total virtual memory usage for the class.

- **i_cl_nvpagehi**: Specifies the virtual memory usage high water mark.

- **i_cl_mem_hwm**: Specifies the maximum number of resident memory pages this class had since its creation (memory high water mark).

- **i_cl_change_level**: Specifies the number of increments each time a change in the current WLM configuration occurs. This field is used by the WLM monitoring tools.
The `wlm_bio_class_info_t` and `wlm_bio_dev_info_t` Structures

Two structures can be used to get the I/O statistics using the `wlm_get_bio_stats` subroutine, depending on whether the application wants per-class or per-device statistics.

The `wlm_bio_class_info_t` structure is used to gather I/O statistics per class and per device. The `wlm_bio_class_info_t` structure contains the following fields:

- **wbc_dev** Specifies the device identifier (`dev_t`)
- **wbc_cid** Specifies the class identifier (index of the class in the kernel `class_control_block` table). Connecting the class ID and the class name can be performed by using the `wlm_get_info` subroutine. This subroutine returns the class name (in the `i_descr` field) and the class ID (in the `i_class_id` field) in the `wlm_info` structure.
- **wbc_regul** Specifies a structure of type `struct wlm_regul`, which contains for the given class and device the following disk I/O statistics:
  - Resource utilization, which is expressed as a percentage of the total available throughput of the device (`consum`)
  - Total number of 512-byte blocks read/written from and to the device by processes in the class since it was created, or since WLM started, whichever happened most recently
- **wbc_delay** Specifies in milliseconds the delay imposed on the I/Os of the processes in the class to the device. This delay is intended to limit utilization by class when it is consuming more than its entitled share.

The `wlm_bio_dev_info_t` structure is used to gather the global statistics for a given device, taking into account all I/Os to and from the device by all the classes accessing the device. This structure contains the following fields:

- **wbd_dev** Specifies the device identifier (`dev_t`)
- **wbd_active_cntrl** Specifies the number of classes actively accessing the device
- **wbd_in_queue** Specifies the number of requests in the device queue
- **wbd_last** Specifies the device statistics for I/Os that occurred during the last second.

This field is an array of integer values. The following symbolic values defined in the `wlm.h` file describe each index in the array:

<table>
<thead>
<tr>
<th>Index</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>WBS_OUT_RTHRPUT</strong></td>
<td>Specifies the number of blocks actually read from the device (I/O completed)</td>
</tr>
<tr>
<td><strong>WBS_OUT_WTHRPUT</strong></td>
<td>Specifies the number of blocks actually written to the device (I/O completed)</td>
</tr>
<tr>
<td><strong>WBS_IN_RTHRPUT</strong></td>
<td>Specifies the requested number of blocks to be read from the device</td>
</tr>
<tr>
<td><strong>WBS_IN_WTHRPUT</strong></td>
<td>Specifies the requested number of blocks to write to the device</td>
</tr>
<tr>
<td><strong>WBS_REQUESTS</strong></td>
<td>Specifies the number of read/write requests</td>
</tr>
<tr>
<td><strong>WBS_QUEUED</strong></td>
<td>Specifies the number of requests queued</td>
</tr>
<tr>
<td><strong>WBS_STARVED</strong></td>
<td>Specifies the number of requests starved (not serviced during the time interval)</td>
</tr>
<tr>
<td><strong>wbd_max</strong></td>
<td>Contains the maximum values observed since the device was first used (after WLM was started) for all the entries of the array being described. For instance, the <code>wbd_max</code> field could contain the maximum number of blocks actually read from the device in one second since the device was first accessed.</td>
</tr>
</tbody>
</table>
wbd_av: Contains the average values for all the entries in the array, such as the average number of requests in the device queue.

wbd_total: Specifies an array of 64-bit integers. This array is parallel to the arrays that, for every entry, contain the total of all the values measured every second since the device was first accessed. For instance, the value could represent the total number of blocks written to the device since the device was first accessed.

**The wlm_proc_info Structure**

The `wlm_proc_info` structure is used to extract Workload Manager information about a process using the `wlm_get_procinfo` subroutine. The `totalconnecttime`, `termtime`, `totalcputime`, `totaldiskio` fields are only meaningful when the total process limits are enabled. The `wlm_proc_info` structure contains the following fields:

- **version**: This field should be initialized with `WLM_VERSION`.

- **wlmflags**: Specifies some Workload Manager properties of the process, such as process with a rset `SWLMRSET` or as tag inheritance on fork `SWLMTAGINHERITFORK` or on exec `SWLMTAGINHERITEXEC`.

- **totalconnecttime**: Specifies the 64-bit number that represents the amount of time (in seconds) for which the login session has been active.

- **totalvmem**: Specifies the total amount of virtual memory used by the process in MBs.

- **termtime**: Specifies the 64-bit number that represents the time (in seconds from 1970) when the process has been requested to terminate.

- **totalcputime**: Specifies the 64-bit number that represents the amount of the CPU consumption (in microseconds) of the process.

- **totaldiskio**: Specifies the 64-bit number that represents the amount of IO (in 512 bytes blocks) the process has run.

- **classname**: Specifies the full name of the superclass or the subclass in which the process is classified.

- **tag**: Specifies the character string associated with the process, if any (see `wlm_set_tag` subroutine).

**Error Codes**

The various WLM API subroutines may return one or several of the following error codes:

- **WLM_ALREADYINIT**: A call to the `wlm_initialize` subroutine has already been made.
- **WLM_ATTERR**: Attribute format error.
- **WLM_ATTGPATTR**: Attribute Value Grouping not allowed in attributes.
- **WLM_ATTGPMISS**: Cannot find Attribute Value Grouping definition.
- **WLM_ATTGPTOOLNG**: Attribute Value Grouping too long.
- **WLM_BADATTAPP**: Could not access file (application field of attributes).
- **WLM_BADATTGP**: Bad format for Attribute Value Grouping.
- **WLM_BADATTGRP**: Unknown group in attributes.
- **WLM_BADATTTAG**: Invalid tag in attributes.
- **WLM_BADATTTYP**: Invalid process type in attributes.
- **WLM_BADATTUSR**: Unknown user in attributes.
WLM_ISCONFSET  This operation cannot apply to a configuration set
WLM_INVRANGE  Invalid time range
WLM_ISLOCKED  WLM configuration is locked: retry the operation later
WLM_LOADERR  A class cannot be loaded into the kernel

WLM_LOCKERR  Cannot lock file
WLM_MANYRULES  Too many assignment rules
WLM_MANYITEMS  Too many items in an assignment rule
WLM_MAXCLASSES  The maximum number of classes has been reached
WLM_MINSMAX  The minimum limit cannot be greater than the soft maximum limit
WLM_MKDIRERR  A directory could not be created
WLM_MULTATTGP  Attribute Value Grouping already defined
WLM_MULTATTR  Multiple specifications not allowed in attributes
WLM_NOADMINSUB  Admin attributes not applicable to subclasses
WLM_NOCLASS  The specified class does not exist
WLM_NOCONFIG  Missing configuration name
WLM_NOCONFINSET  No configuration in the set
WLM_NOCONNECT  Failure to connect to WLM daemon
WLM_NODAEMON  Failure to start WLM daemon
WLM_NOMEM  Not enough memory
WLM_NOSHRLIM  Cannot specify a total memory limit for Shared class
WLM_NOSHRULE  Cannot specify rule for Shared class
WLM_NOSUBS  The target superclass has no subclasses
WLM_NOSYSLIM  Cannot specify a total memory limit for System class
WLM_NOSYSMAX  Hardmax not allowed on memory for System class
WLM_NOTASSGND  Process is not assigned
WLM_NOTCOMPLETE  Could not assign all processes (warning)
WLM_NOTCURRENT  Superclass update only applies to current configuration
WLM_NOTINITED  No prior call to the wlm_initialize subroutine
WLM_NOTRUNNING  WLM is not running
WLM_NOWILDCRD  Wildcards not allowed in this field
WLM_ONEDEFAULT  Only one default time range allowed in a set
WLM_OPENERR  A file could not be opened
WLM_QUERYERR  Cannot query WLM state
WLM_READERR  Cannot read file
WLM_REFRULE  A class is still referred to by rules
WLM_REMERR  An attempt to remove a file did not succeed
WLM_RENAMERR  An attempt to rename a file did not succeed
WLM_RMPREDEF  Predefined classes (such as Default and System) cannot be removed
WLM_RUNCLASS  A class specified in the rules file does not exist

WLM_RSVVDNAME  Predefined classes cannot be redefined
WLM_RULERR  An assignment rule has an invalid format
WLM_RULESERR  The assignment rules table cannot be loaded into the kernel
WLM_RULTOOLNG  Rule too long
WLM_RUNERR  The WLM library was not able to execute a command needed for the specific
function. This is not an application error, but most likely a system
administration problem. The commands used by the library are basic
operating system commands such as the lsuser, lsgroup, echo, and grep
commands.

WLM_RUNERRATT  Cannot expand attribute
WLM_SETERR  The WLM state transition requested is illegal
WLM_SHAREDLIM  Shared class can have shares and limits set only for memory
WLM_SHAREDSUB  Shared superclass cannot have subclasses
WLM_SMAXHMAX  The soft maximum limit cannot be greater than the hard maximum limit
WLM_STATERR
One or more file name(s) specified in the application field of an assignment rule could not be accessed. The corresponding name(s) are ignored (warning)

WLM_SUBINVALID
No subclass specification allowed for this operation

WLM_SUMMINS
The sum that the minimum limits for a given resource and a given tier cannot exceed 100%

WLM_SYMLERR
An attempt to create a symbolic link did not succeed

WLM_TAGTOOLONG
Tag is too long

WLM_TOOLONG
The specified class name is too long

WLM_TOOLONGATT
Attribute list too long

WLM_TOOMANYATT
Too many items in attributes

WLM_TOOMANYPID
Process ID list too long

WLM_TOOSMALL
Output buffer too small

WLM_TOTALLIMITOUTOFRANGE
Invalid total limit (outside allowed range)

WLM_TRGAPS
Gaps between time ranges in a configuration set

WLM_TRINDEFAULT
Time range not allowed in default stanza

WLM_TROVERLAP
Time ranges overlap in a configuration set

WLM_UNLOADERR
Cannot unload class

WLM_UNSUPP
Operation or flags value not supported

WLM_WILDCRDATT
Wildcards not allowed in this attribute field

WLM_WRITERR
An attempt to write to a file did not succeed

Related Information
The wlm_init_class_definition subroutine, wlm_assign subroutine, wlm_get_info subroutine.

Workload management in Operating system and device management

x25sdefs.h File for X.25

Purpose
Contains the structures used by the X.25 application programming interface (API).

Description
The /usr/include/x25sdefs.h file defines the following structures used by X.25 subroutines.

Miscellaneous Structures

- **cb_link_name_struct**
  Used to indicate the name of the X.25 port.

- **cb_msg_struct**
  Used to indicate the type of message being received.

- **ctr_array_struct**
  Used to store the counter values and identifiers for use with the x25_ctr_wait structure.

Structures Used to Establish Calls and Transfer Data

- **cb_call_struct**
  Used for calls made and accepted.

- **cb_data_struct**
  Used for data transferred during a call.

- **cb_fac_struct**
  Used for information about optional facilities being used.

- **cb_pvc_alloc_struct**
  Used to indicate the logical channel number and port assigned to a permanent virtual circuit (PVC).

Structures Used to Clear, Interrupt and Reset Calls

- **cb_clear_struct**
  Used for calls being cleared.
cb_int_data_struct
Used for data sent or received in an interrupt packet.
cb_res_struct
Used for data sent or received in a reset-request packet.

**Structures Used to Manage X.25 Communications**

- **cb_circuit_info_struct**
  Used for information about a virtual circuit.
- **cb_link_stats_struct, x25_query_data, and x25_stats**
  Used for information about an X.25 adapter.
- **cb_link_stats_struct**
  Used for statistics for an X.25 port.

For more information, see the individual descriptions of these structures.

**Related Information**
Using the X.25 Subroutines in *AIX 5L Version 5.3 Communications Programming Concepts*.

---

**cb_call_struct Structure for X.25**

**Purpose**
Used by the `x25_call`, `x25_call_accept`, and `x25_receive` subroutines to pass the X.25 port name, called and calling addresses, facilities, and user data.

**Syntax**
```c
#define X25FLG_D_BIT 0x00000001
#define X25FLG_LINK_NAME 0x00000002
#define X25FLG_CALLED_ADDR 0x00000004
#define X25FLG_CALLING_ADDR 0x00000008
#define X25FLG_CB_FAC 0x00000010
#define X25FLG_USER_DATA 0x00000020

struct cb_call_struct
{
    unsigned long flags;
    char *link_name;
    char *called_addr;
    char *calling_addr;
    struct cb_fac_struct *cb_fac;
    int user_data_len;
    unsigned char *user_data;
};
```

**Flags**

- **X25_FLG_D_BIT**
  Indicates that the call uses D-bit procedures.
- **X25_FLG_LINK_NAME**
  Indicates that the `link_name` field is used.
- **X25_FLG_CALLED_ADDR**
  Indicates that the `called_addr` field is used.
- **X25_FLG_CALLING_ADDR**
  Indicates that the `calling_addr` field is used.
- **X25_FLG_CB_FAC**
  Indicates that the `cb_fac` field is used.
- **X25_FLG_USER_DATA**
  Indicates that the `user_data` field is used.

**Fields**

- **flags**
  Notification to the API that the associated field has been used.
link_name
Name of the X.25 port used for an incoming call.

Note: This is set to null on received packets.
called_addr
Pointer to the network user address (NUA) of the called data terminal equipment (DTE).
The address is given in ASCIIZ format.
calling_addr
Pointer to the NUA of the calling DTE. The address is given in ASCIIZ format.
cb_fac
Pointer to the facilities information in the cb_fac_struct structure.
user_data_len
Field length for call user data (CUD).
user_data
Pointer to call user data (CUD).

---

cb_circuit_info_struct Structure for X.25

Purpose
Used by the x25_circuit_query subroutine to return information about the circuit.

Syntax
```c
#define X25FLG_INCOMING_PACKET_SIZE 0x00000001
#define X25FLG_OUTGOING_PACKET_SIZE 0x00000002
#define X25FLG_INCOMING_THROUGHPUT_CLASS 0x00000004
#define X25FLG_OUTGOING_THROUGHPUT_CLASS 0x00000008
#define X25FLG_INCOMING_WINDOW_SIZE 0x00000010
#define X25FLG_OUTGOING_WINDOW_SIZE 0x00000020
```

```c
struct cb_circuit_info_struct
{
    unsigned long flags;
    unsigned short lcn;
    unsigned int incoming_packet_size;
    unsigned int outgoing_packet_size;
    unsigned int incoming_throughput_class;
    unsigned int outgoing_throughput_class;
    unsigned int incoming_window_size;
    unsigned int outgoing_window_size;
};
```

Flags

**X25_FLG_INCOMING_PACKET_SIZE**
Indicates that the incoming_packet_size field is used.

**X25_FLG_OUTGOING_PACKET_SIZE**
Indicates that the outgoing_packet_size field is used.

**X25_FLG_INCOMING_THROUGHPUT_CLASS**
Indicates that the incoming_throughput_class field is used.

**X25_FLG_OUTGOING_THROUGHPUT_CLASS**
Indicates that the outgoing_throughput_class field is used.

**X25_FLG_INCOMING_WINDOW_SIZE**
Indicates that the incoming_window_size field is used.

**X25_FLG_OUTGOING_WINDOW_SIZE**
Indicates that the outgoing_window_size field is used.
Fields

flags
lcn
incoming_packet_size
outgoing_packet_size
incoming_throughput_class
outgoing_throughput_class
incoming_window_size
outgoing_window_size

Notification to the API that the associated field has been used.
Logical channel number.
Actual size for incoming packets.
Actual size for outgoing packets.
Throughput class for incoming calls.
Throughput class for outgoing calls.
Number of incoming packets that can be sent without confirmation.
Number of outgoing packets that can be sent without confirmation.

cb_clear_struct Structure for X.25

Purpose
Used by the x25_call_clear and x25_receive subroutines to pass the clear cause and diagnostic values, called and calling addresses, facilities information, and user data.

Syntax

```
#define X25FLG_CAUSE          0x00000001
#define X25FLG_DIAGNOSTIC    0x00000002
#define X25FLG_CALLED_ADDR   0x00000004
#define X25FLG_CALLING_ADDR  0x00000008
#define X25FLG_CB_FAC        0x00000010
#define X25FLG_USER_DATA     0x00000020
```

```
struct cb_clear_struct
{
    unsigned long flags;
    u_char cause;
    u_char diagnostic;
    char *called_addr;
    char *calling_addr;
    struct cb_fac_struct *cb_fac;
    int user_data_len;
    u_char *user_data;
};
```

Flags

<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X25_FLG_CAUSE</td>
<td>Indicates that the cause field is used.</td>
</tr>
<tr>
<td>X25_FLG_DIAGNOSTIC</td>
<td>Indicates that the diagnostic field is used.</td>
</tr>
<tr>
<td>X25_FLG_CALLED_ADDR</td>
<td>Indicates that the called_addr field is used.</td>
</tr>
<tr>
<td>X25_FLG_CALLING_ADDR</td>
<td>Indicates that the calling_addr field is used.</td>
</tr>
<tr>
<td>X25_FLG_CB_FAC</td>
<td>Indicates that the cb_fac field is used.</td>
</tr>
<tr>
<td>X25_FLG_USER_DATA</td>
<td>Indicates that the user_data field is used.</td>
</tr>
</tbody>
</table>

Fields

flags
cause
diagnostic
called_addr
calling_addr

Notification to the API that the associated field has been used.
Cause value to be inserted in clear packet.
Diagnostic reason to be inserted in packet.
Pointer to the network user address (NUA) of the called data terminal equipment (DTE).
The address is given in ASCII format.
Pointer to the NUA of the calling DTE. The address is given in ASCII format.
cb_fac
user_data_len
user_data

Pointer to the facilities information in the `cb_fac_struct` structure.
Length of user-data field.
Pointer to user data. This can be used only if "fast select" has been requested in the call-request packet.

---

**cb_data_struct Structure for X.25**

**Purpose**
Used by the `x25_send` and `x25_receive` subroutines to pass data control information.

**Syntax**

```c
#define X25FLG_D_BIT 0x00000001
#define X25FLG_Q_BIT 0x00000002
#define X25FLG_M_BIT 0x00000004
#define X25FLG_DATA 0x00000008

struct cb_data_struct
{
    unsigned long flags;
    int data_len;
    unsigned char *data;
};
```

**Flags**

- **X25FLG_D_BIT**  If the D-bit has been set in the call packet, and the value is not zero, the remote data terminal equipment (DTE) must acknowledge the packet.
- **X25FLG_Q_BIT**  Sets the Q-bit in the packet. A nonzero value is converted to a single 1-bit in the packet.
- **X25FLG_M_BIT**  Sets the M-bit in the packet. A nonzero value is converted to a single 1-bit in the packet.
- **X25_FLG_DATA**  Indicates that the data field is used.

**Fields**

- **flags**  Notification to the API that the associated field has been used.
- **data_len**  Length of data.
- **data**  Pointer to actual data.

---

**cb_dev_info_struct Structure for X.25**

**Purpose**
Used by the `x25_device_query` subroutine to pass device information.

**Syntax**

```c
#define X25FLG_NUA 0x00000001
#define X25FLG_NO_OF_VCS 0x00000002
#define X25FLG_MAX_RX_PACKET_SIZE 0x00000004
#define X25FLG_MAX_TX_PACKET_SIZE 0x00000008
#define X25FLG_DEFAULT_SVC_RX_PACKET_SIZE 0x00000010
#define X25FLG_DEFAULT_SVC_TX_PACKET_SIZE 0x00000020

struct cb_dev_info_struct
```
{ unsigned long flags;
  char *nua;
  unsigned int no_of_vcs;
  unsigned int max_rx_packet_size;
  unsigned int max_tx_packet_size;
  unsigned int default_svc_rx_packet_size;
  unsigned int default_svc_tx_packet_size;
};

**Flags**

- **X25_FLG_NUA**
  Indicates that the `nua` field is used.

- **X25_FLG_NO_OF_VCS**
  Indicates that the `no_of_vcs` field is used.

- **X25_FLG_MAX_RX_PACKET_SIZE**
  Indicates that the `max_rx_packet_size` field is used.

- **X25_FLG_MAX_TX_PACKET_SIZE**
  Indicates that the `max_tx_packet_size` field is used.

- **X25_FLG_DEFAULT_SVC_RX_PACKET_SIZE**
  Indicates that the `default_svc_rx_packet_size` field is used.

- **X25_FLG_DEFAULT_SVC_TX_PACKET_SIZE**
  Indicates that the `default_svc_tx_packet_size` field is used.

**Fields**

- **flags**
  Notification to the API that the associated field has been used.

- **nua**
  Pointer to the network user address (NUA) recorded for the device in ASCII format.

- **no_of_vcs**
  Number of permanent virtual circuits (PVCs) configured on this device.

- **max_rx_packet_size**
  Maximum receive packet size in bytes.

- **max_tx_packet_size**
  Maximum transmit packet size in bytes.

- **default_svc_rx_packet_size**
  Default receive packet size in bytes.

- **default_svc_tx_packet_size**
  Default transmit packet size in bytes.

---

**cb_fac_struct Structure for X.25**

**Purpose**

Used by the `x25_call` and `x25_call_accept` subroutines to pass facilities information.

**Syntax**

```c
#define X25FLG_RPOA 0x00000001
#define X25FLG_PSIZ 0x00000002
#define X25FLG_WSIZ 0x00000004
#define X25FLG_TCLS 0x00000008
#define X25FLG_REV_CHRG 0x00000010
#define X25FLG_FASTSEL 0x00000000
```
```c
#define X25FLG_FASTSEL_RSP 0x00000040
#define X25FLG_CUG 0x00000080
#define X25FLG_OA_CUG 0x00000100
#define X25FLG_BI_CUG 0x00000200
#define X25FLG_NUI_DATA 0x00000400
#define X25FLG_CI_SEG_CNT 0x00000800
#define X25FLG_CI_MON_UNT 0x00001000
#define X25FLG_CI_CALL_DUR 0x00002000
#define X25FLG_CI_REQUEST 0x00004000
#define X25FLG_CLAMN 0x00008000
#define X25FLG_CALL_REDR 0x00010000
#define X25FLG_TRAN_DEL 0x00020000
#define X25FLG_CALLING_ADDR_EXT 0x00040000
#define X25FLG_CALLED_ADDR_EXT 0x00080000
#define X25FLG_MIN_TCLS 0x00100000
#define X25FLG_END_TO_END_DEL 0x00200000
#define X25FLG_EXP_DATA 0x00400000
#define X25FLG_FACEXT 0x00800000

struct cb_fac_struct
{
    u_long flags;
    unsigned fac_ext_len;
    u_char *fac_ext;        /*
     *for non-X.25 facilities */
    u_char psiz_clg;
    u_char psiz_cld;
    u_char wsiz_clg;
    u_char wsiz_cld;
    u_char tcls_clg;
    u_char tcls_cld;
    unsigned rpoa_id_len;
    ushort *rpoa_id;
    ushort cug_id;
    unsigned nui_data_len;
    u_char *nui_data;
    unsigned ci_seg_cnt_len;
    u_char *ci_seg_cnt;
    unsigned ci_mon_unt_len;
    u_char *ci_mon_unt;
    unsigned ci_cal_dur_len;
    u_char *ci_cal_dur;
    u_char call_redr_addr[X25_MAX_ASCII_ADDRESS_LENGTH];
    u_char call_redr_reason;
    short tran_del;
    u_char calling_addr_ext_use;
    char calling_addr_ext[X25_MAX_EXT_ADDR_DIGITS+1];
    u_char called_addr_ext_use;
    char called_addr_ext[X25_MAX_EXT_ADDR_DIGITS+1];
    u_char clamn;
    u_char min_tcls_clg;
    u_char min_tcls_cld;
    unsigned end_to_end_del_len;
    ushort end_to_end_del[3];
};

Note: The example shows how to code the cb_fac_struct structure.
```
Flags

X25FLG_RPOA  Recognized private operating agency selection required (rpoa_id).
X25FLG_PSZ   Packet size selection (psiz_clg, psiz_clid).
X25FLG_WSIZ  Window size selection (wsiz_clg, wsiz_clid).
X25FLG_TCLS  Throughput class required (tcls_clg, tcls_clid).
X25FLG_REV_CHRG Reverse Charge required (no corresponding field).
X25FLG_FASTSEL Fast select (no corresponding field).
X25FLG_FASTSEL_RSP Indicates whether a restricted response is required when the
                  X25FLG_FASTSEL flag is also requested (no corresponding field).
X25FLG_CUG   Closed user group selection required (cug_id).
X25FLG_OA_CUG Closed user group with outgoing access (basic format) selection
                  required (cug_id).
X25FLG_BI_CUG Bilateral closed user group selection required (cug_id).
X25FLG_CI_SEG_CNT Charging information: segment count (ci_seg_cnt).
X25FLG_CI_MON_UNT Charging information: monetary unit (ci_mon_unt).
X25FLG_CI_CAL_DUR Charging information: call duration (ci_cal_dur).
X25FLG_CI_REQUEST Charging information is required (no corresponding field).
X25FLG_CLAMN  Called line address modified notification (clamn).
X25FLG_CALL_REDR Call redirection notification (call_redr_addr, call_redr_reason).
X25FLG_TRAN_DEL Transit delay selection and notification (tran_del).
X25FLG_CALLING_ADDR_EXT Calling address extension (calling_addr_ext_use, calling_addr_ext).
X25FLG_CALLED_ADDR_EXT Called address extension (called_addr_ext_use, called_addr_ext).
X25FLG_MIN_TCLS Quality of service negotiation: minimum throughput class
                  (min_tcls_clg, min_tcls_clid).
X25FLG_END_TO_END_DEL Quality of service negotiation: end-to-end transit delay (end_to_end_del).
X25FLG_EXP_DATA Expedited data negotiation (no corresponding field).
X25FLG_FACEXT Facilities extension: for all other facilities, including national options (fac_ext).

Fields

This section explains the meanings of structure fields but not the lengths associated with individual pointer fields.

flags
fac_ext

Notification to the API that the associated field has been used.

Pointer to the facilities extension array (extra facility information provided by
the user or network). No checking is made on the validity of this information.
It allows extra facilities that the main cb_fac structure does not include. The
elements of the fac_ext field are copied directly into the facility field.

When the information is provided by the X.25 network or by the remote data
terminal equipment (DTE), it is the responsibility of the application to interpret
the field.

Only elements up to the first non-X.25 facility are decoded by the API. Facility
markers must be used in the fac_ext field if such facilities are required.
psiz_clg

Indicates the requested size for packets transmitted from the calling DTE. The following are supported values:

- 0x04 = 16 octets
- 0x05 = 32 octets
- 0x06 = 64 octets
- 0x07 = 128 octets
- 0x08 = 256 octets
- 0x09 = 512 octets
- 0x0A = 1024 octets
- 0x0B = 2048 octets
- 0x0C = 4096 octets

psiz_cld

Requested size for packets transmitted from the called DTE. Supported values are the same as for the psiz_clg field.

wsiz_clg

Requested size for the window for packets transmitted by the calling DTE. Values range from 0x01 to 0x07 inclusive.

wsiz_cld

Requested size for the window for packets to be transmitted by the called DTE. Values range from 0x01 to 0x07 inclusive.

tcls_clg

Throughput class requested for data to be sent by the calling DTE. The following are supported values:

- 0x07 = 1200 bits per second
- 0x08 = 2400 bits per second
- 0x09 = 4800 bits per second
- 0x0A = 9600 bits per second
- 0x0B = 19,200 bits per second
- 0x0C = 48,000 bits per second

wsiz_clg

Throughput class request for data sent from the called DTE. Supported values are the same as for the tcls_clg field.

rpoa_id

Indicates the requested RPOA (Requested Private Operating Agency) transit network. Each element of the array is an RPOA identifier.

cug_id

Indicates the identifier of a closed user group (CUG). Used for all modes of CUG and also for bilateral CUGs.

nui_data

Network user identification data in a format specified by the network administrator.

ci_seg_cnt

Charging information: segment count data.

ci_mon_unt

Charging information: monetary unit data.

ci_cal_dur

Charging information: call-duration data.

call_redr_addr

The address to which the call has been redirected. The address is stored in ASCIIZ format.

call_redr_reason

Contains reason for call redirection.

tran_del

Transit delay in milliseconds.

calling_addr_ext_use

Indicates the use of the calling address extension.

calling_addr_ext

Up to 40 digits containing the calling address extension. The address extension is stored in ASCIIZ format. The following are values for the extended calling and called address flags:

- #DEFINE X25_FAC_ADDR_EXT_USE_ENTIRE_OSI_NSAP(0)
- #DEFINE X25_FAC_ADDR_EXT_USE_PARTIAL_OSI_NSAP(1)
- #DEFINE X25_FAC_ADDR_EXT_USE_NON_OSI(2)

called_addr_ext_use

Indicates the use of the called address extension.

called_addr_ext

Up to 40 digits containing the called address extension. The address extension is stored in ASCIIZ format. See the calling_addr_ext field for values.

clamn

Called line address modified notification. Contains the reason for redirection.
min_tcls_clg  Throughput class requested for data to be sent by the calling DTE. The following are supported values:
  • 0x07 = 1200 bits per second
  • 0x08 = 2400 bits per second
  • 0x09 = 4800 bits per second
  • 0x0A = 9600 bits per second
  • 0x0B = 19,200 bits per second
  • 0x0C = 48,000 bits per second

min_tcls_cld  Throughput class request for data sent from the called DTE. Supported values are the same as for the min_tcls_clg field.

end_to_end_del  Specifies cumulative requested end-to-end and maximum-acceptable transit delays. Requested end-to-end and maximum-acceptable values are optional.

Examples
This is a simple example of the use of the cb_fac_struct structure:

/*

 &
*/

struct cb_call_struct cb_call;
struct cb_fac_struct fac_struct;
u_char facilities_extension[10], facilities_extension[8];
ushort rpoa_ext_id[3] = {7, 8, 9};
char extended_calling_addr[]= "1234567890"; /* extension */
/* Initialize flags

  */
fac_struct.flags = 0;
/* Use of RPOAE

  */
fac_struct.rpoa_id = rpoa_ext_id;
fac_struct.rpoa_id_len = 3;
fac_struct.flags |= X25FLG_RPOA;
/* Use of extended addressing

  */
fac_struct.calling_addr_ext = extended_calling_addr;
fac_struct.flags |= X25FLG_CALLING_ADDR_EXT;
/* Use of extended facilities

  */
facilities_extension[0] = 0x00; /*
start of a Facility Marker */
facilities_extension[1] = 0x00; /*
non_X25 facility supported */

  */
/* by calling DTE
  */
facilities_extension[2] = 0x55; /*
a facility */
*/
facilities_extension[3] = 0x66;    /* a facility */
facilities_extension[4] = 0x00;    /* start of a Facility Marker */
facilities_extension[5] = 0xFF;    /* non_X25 facility supported */
facilities_extension[6] = 0x88;    /* a facility */
facilities_extension[7] = 0x99;    /* a facility */

```c
strcpy(fac_struct.fac_ext, facilities_extension);
fac_struct.fac_ext_len = 8;
fac_struct.flags |= X25FLG_FACEXT;
```

```c
cb_call.cb_fac = &fac_struct;
cb_call.flags = X25FLG_CB_FAC;
```

---

**cb_int_data_struct Structure for X.25**

**Purpose**

*Used by the x25_interrupt and x25_receive subroutines to pass the interrupt data.*

**Syntax**

```c
#define X25FLG_INT_DATA 0x00000001

struct cb_int_struct {
    unsigned long flags;
    unsigned char int_data_len;
    unsigned char *int_data;
};
```

**Flags**

X25FLG_INT_DATA  
A non-zero value indicates the presence of data in the cb_int_data structure.

**Fields**

- **flags**  
  Notification to the API that the associated field has been used.
- **int_data_len**  
  Length of data in the cb_int_data structure.
- **int_data**  
  Interrupt data.
cb_link_name_struct Structure for X.25

**Purpose**

Used by the X25_init, x25_link_connect, x25_link_disconnect, x25_link_monitor, x25_device_query, and x25_term subroutines to pass the name of the X.25 port.

**Syntax**

```c
#define X25FLG_LINK_NAME 0x00000002

struct cb_link_name_struct
{
    unsigned long flags;
    char *link_name;
};
```

**Flags**

- **X25_FLG_LINK_NAME** Indicates that the link_name field is used.

**Fields**

- **flags** Notification to the API that the associated field has been used.
- **link_name** Name of the X.25 port.

---

cb_link_stats_struct, x25_query_data, or x25_stats Structure for X.25

**cb_link_stats_struct Structure**

Used by the x25_link_statistics subroutine to pass statistics about an X.25 port.

```c
#define X25FLG_NO_OF_VCS 0x00000008
#define X25FLG_LINK_STATS 0x00000020

struct cb_link_stats_struct
{
    unsigned long flags;
    unsigned int no_of_vcs;
    struct x25_query_data x25_stats;
};
```

**Flags**

- **X25_FLG_NO_OF_VCS** Indicates that the no_of_vcs field is used.
- **X25_FLG_LINK_STATS** Indicates that the x25_stats structure is being used.

**Fields**

- **flags** Notification to the API that the associated field has been used
- **no_of_vcs** Number of virtual circuits currently in use for the X.25 port specified
- **x25_stats** Pointer to an x25_query_data structure containing CIO and X.25 statistics
x25_query_data Structure

The **x25_query_data** structure is returned from the **CIO_QUERY** ioctl operation. It includes two structures: the **cio_stats** structure containing standard statistics values found in the **sys/comio.h** file, and the **x25_stats** structure containing specific X.25 statistics.

```c
struct x25_query_data {
    struct cio_stats cc;
    struct x25_stats ds;
};
```

x25_stats Structure

The **x25_stats** structure contains specific X.25 statistics.

**Note:** Flags are not used with this structure.

```c
typedef unsigned short x25_stat_value_t;
struct x25_stats {
    x25_stat_value_t ignored_f_tx;
    x25_stat_value_t rr_f_tx;
    x25_stat_value_t rnr_f_tx;
    x25_stat_value_t rej_f_tx;
    x25_stat_value_t info_f_tx;
    x25_stat_value_t sabm_f_tx;
    x25_stat_value_t disc_f_tx;
    x25_stat_value_t ua_f_tx;
    x25_stat_value_t frmr_f_tx;
    x25_stat_value_t bad_nr_f_tx;
    x25_stat_value_t unknown_f_tx;
    x25_stat_value_t xid_f_tx;
    x25_stat_value_t bad_length_f_tx;
    x25_stat_value_t bad_length_f_rx;
    x25_stat_value_t unknown_f_rx;
    x25_stat_value_t xid_f_rx;
    x25_stat_value_t data_p_tx;
    x25_stat_value_t rr_p_tx;
    x25_stat_value_t rnr_p_tx;
    x25_stat_value_t interrupt_p_tx;
    x25_stat_value_t interrupt_confirm_p_tx;
    x25_stat_value_t call_request_p_tx;
    x25_stat_value_t call_accept_p_tx;
    x25_stat_value_t clear_request_p_tx;
    x25_stat_value_t clear_confirm_p_tx;
    x25_stat_value_t carrier_loss;
    x25_stat_value_t connect_time; /* In seconds */
    x25_stat_value_t t4_expirations;
    x25_stat_value_t t4_n2_times;
    x25_stat_value_t ignored_f_rx;
    x25_stat_value_t rr_f_rx;
    x25_stat_value_t rnr_f_rx;
    x25_stat_value_t rej_f_rx;
    x25_stat_value_t info_f_rx;
    x25_stat_value_t sabm_f_rx;
    x25_stat_value_t disc_f_rx;
    x25_stat_value_t ua_f_rx;
    x25_stat_value_t frmr_f_rx;
    x25_stat_value_t bad_nr_f_rx;
    x25_stat_value_t unknown_f_rx;
    x25_stat_value_t xid_f_rx;
    x25_stat_value_t bad_length_f_rx;
    x25_stat_value_t carrier_loss;
    x25_stat_value_t connect_time; /* In seconds */
};
```
x25_stat_value_t reset_request_p_tx;

x25_stat_value_t reset_confirm_p_tx;

x25_stat_value_t diagnostic_p_tx;

x25_stat_value_t registration_p_tx;

x25_stat_value_t registration_confirm_p_tx;

x25_stat_value_t restart_p;

x25_stat_value_t restart_confirm_p;

x25_stat_value_t error_p;

x25_stat_value_t t20_expirations;

x25_stat_value_t t21_expirations;

x25_stat_value_t t22_expirations;

x25_stat_value_t t23_expirations;

x25_stat_value_t vc_establishments;

x25_stat_value_t t24_expirations;

x25_stat_value_t t25_expirations;

x25_stat_value_t t26_expirations;

x25_stat_value_t t28_expirations;

x25_stat_value_t data_p_rx;

x25_stat_value_t rr_p_rx;

x25_stat_value_t rnr_p_rx;

x25_stat_value_t interrupt_p_rx;

x25_stat_value_t interrupt_confirm_p;

x25_stat_value_t incoming_call_p_rx;

x25_stat_value_t call_connected_p_rx;

x25_stat_value_t clear_indication_p;

x25_stat_value_t clear_confirm_p;

x25_stat_value_t reset_indication_p;

x25_stat_value_t reset_confirm_p;

x25_stat_value_t diagnostic_p;

x25_stat_value_t registration_p;

x25_stat_value_t registration_confirm_p;

x25_stat_value_t restart_p;

x25_stat_value_t restart_confirm_p;

int transmit_profile [16];

int receive_profile [16];

};

Fields

ignored_f_tx

rr_f_tx

rnr_f_tx

rej_f;

info_f;

sabm_f;

sarm_dm_f;

disc_f;

ua_f;

frmr_f;

bad_nr_f;

unknown_f;

xml_f;

bad_length_f;

tl_expirations;

lv12_connects;

lv12_disconnects;

carrier_loss;

connect_time;

t4_expirations;

t4_n2_expirations

Number of transmitted frames that have been ignored instead of being transmitted.

Number of RR (receive ready) frames transmitted.

Number of RNR (receive not ready) frames transmitted.

Number of REJ (reject) frames transmitted.

Number of INFO (information) frames transmitted.

Number of SABM (set asynchronous balanced mode) frames transmitted.

Number of SARM/DM frames transmitted.

Number of DISC (disconnect) frames transmitted.

Number of UA (unnumbered acknowledgment) frames transmitted.

Number of FRMR (frame received) frames transmitted.

Number of frames transmitted with a bad N(R) value.

Number of unknown frames transmitted.

Number of XID frames transmitted.

Number of bad length frames transmitted.

Number of times the T1 timer has timed out.

Number of times the frame level has been connected.

Number of times the frame level has been disconnected.

Number of times the carrier signal was lost.

Number of seconds that the link has been connected.

Number of times the T4 timer has timed out.

Number of times the T4 timer has timed out N2 times.
<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ignored_f_rx</td>
<td>Number of received frames that have been ignored instead of being received.</td>
</tr>
<tr>
<td>rr_f_rx</td>
<td>Number of RR frames received.</td>
</tr>
<tr>
<td>rnr_f_rx</td>
<td>Number of RNR frames received.</td>
</tr>
<tr>
<td>rej_f_rx</td>
<td>Number of REJ frames received.</td>
</tr>
<tr>
<td>info_f_rx</td>
<td>Number of INFO frames received.</td>
</tr>
<tr>
<td>sabm_f_rx</td>
<td>Number of SABM frames received.</td>
</tr>
<tr>
<td>sarm_dm_f_rx</td>
<td>Number of SARM/DM frames received.</td>
</tr>
<tr>
<td>disc_f_rx</td>
<td>Number of DISC frames received.</td>
</tr>
<tr>
<td>ua_f_rx</td>
<td>Number of UA frames received.</td>
</tr>
<tr>
<td>frmr_f_rx</td>
<td>Number of FRMR frames received.</td>
</tr>
<tr>
<td>bad_nr_f_rx</td>
<td>Number of frames received with a bad N(R) value.</td>
</tr>
<tr>
<td>unknown_f_rx</td>
<td>Number of unknown frames received.</td>
</tr>
<tr>
<td>xid_f_rx</td>
<td>Number of XID frames received.</td>
</tr>
<tr>
<td>bad_length_f_rx</td>
<td>Number of bad length frames received.</td>
</tr>
<tr>
<td>data_p_tx</td>
<td>Number of data packets transmitted.</td>
</tr>
<tr>
<td>rr_p_tx</td>
<td>Number of RR packets transmitted.</td>
</tr>
<tr>
<td>rnr_p_tx</td>
<td>Number of RNR packets transmitted.</td>
</tr>
<tr>
<td>interrupt_p_tx</td>
<td>Number of interrupt packets transmitted.</td>
</tr>
<tr>
<td>interrupt_confirm_p_tx</td>
<td>Number of interrupt-confirmation packets transmitted.</td>
</tr>
<tr>
<td>call_request_p_tx</td>
<td>Number of call-request packets transmitted.</td>
</tr>
<tr>
<td>call_accept_p_tx</td>
<td>Number of call-accept packets transmitted.</td>
</tr>
<tr>
<td>clear_request_p_tx</td>
<td>Number of clear-request packets transmitted.</td>
</tr>
<tr>
<td>clear_confirm_p_tx</td>
<td>Number of clear-confirm packets transmitted.</td>
</tr>
<tr>
<td>reset_request_p_tx</td>
<td>Number of reset-request packets transmitted.</td>
</tr>
<tr>
<td>reset_confirm_p_tx</td>
<td>Number of reset-confirm packets transmitted.</td>
</tr>
<tr>
<td>diagnostic_p_tx</td>
<td>Number of diagnostic packets transmitted.</td>
</tr>
<tr>
<td>registration_p_tx</td>
<td>Number of registration packets transmitted.</td>
</tr>
<tr>
<td>registration_confirm_p_tx</td>
<td>Number of registration-confirmation packets transmitted.</td>
</tr>
<tr>
<td>restart_p_tx</td>
<td>Number of restart packets transmitted.</td>
</tr>
<tr>
<td>restart_confirm_p_tx</td>
<td>Number of restart-confirmation packets transmitted.</td>
</tr>
<tr>
<td>error_p_tx</td>
<td>Number of error packets transmitted.</td>
</tr>
<tr>
<td>t20_expirations</td>
<td>Number of times the T20 timer has timed out.</td>
</tr>
<tr>
<td>t21_expirations</td>
<td>Number of times the T21 timer has timed out.</td>
</tr>
<tr>
<td>t22_expirations</td>
<td>Number of times the T22 timer has timed out.</td>
</tr>
<tr>
<td>t23_expirations</td>
<td>Number of times the T23 timer has timed out.</td>
</tr>
<tr>
<td>vc_establishments</td>
<td>Number of times a virtual circuit has been established.</td>
</tr>
<tr>
<td>t24_expirations</td>
<td>Number of times the T24 timer has timed out.</td>
</tr>
<tr>
<td>t25_expirations</td>
<td>Number of times the T25 timer has timed out.</td>
</tr>
<tr>
<td>t26_expirations</td>
<td>Number of times the T26 timer has timed out.</td>
</tr>
<tr>
<td>t28_expirations</td>
<td>Number of times the T28 timer has timed out.</td>
</tr>
<tr>
<td>data_p_rx</td>
<td>Number of data packets received.</td>
</tr>
<tr>
<td>rr_p_rx</td>
<td>Number of RR packets received.</td>
</tr>
<tr>
<td>rnr_p_rx</td>
<td>Number of RNR packets received.</td>
</tr>
<tr>
<td>interrupt_p_rx</td>
<td>Number of interrupt packets received.</td>
</tr>
<tr>
<td>interrupt_confirm_p_rx</td>
<td>Number of interrupt-confirmation packets received.</td>
</tr>
<tr>
<td>call_request_p_rx</td>
<td>Number of call-request packets received.</td>
</tr>
<tr>
<td>call_accept_p_rx</td>
<td>Number of call-accept packets received.</td>
</tr>
<tr>
<td>clear_request_p_rx</td>
<td>Number of clear-request packets received.</td>
</tr>
<tr>
<td>clear_confirm_p_rx</td>
<td>Number of clear-confirm packets received.</td>
</tr>
<tr>
<td>reset_request_p_rx</td>
<td>Number of reset-request packets received.</td>
</tr>
<tr>
<td>reset_confirm_p_rx</td>
<td>Number of reset-confirm packets received.</td>
</tr>
<tr>
<td>diagnostic_p_rx</td>
<td>Number of diagnostic packets received.</td>
</tr>
<tr>
<td>registration_p_rx</td>
<td>Number of registration packets received.</td>
</tr>
</tbody>
</table>
registration_confirm_p_rx
Number of registration-confirmation packets received.

restart_p_rx
Number of restart packets received.

restart_confirm_p_rx
Number of restart-confirmation packets received.

receive_profile[16]
A profile of the receive packet sizes in use on this X.25 port. Each element of the array contains a count of the number of packets received, since the X.25 adapter was last configured, whose sizes are in the range specified. See the transmit_profile field for a list of these size values.

transmit_profile[16]
A profile of the transmission packet sizes used on this X.25 port. Each element of the array contains a count of the number of packets sent, since the X.25 adapter was last configured, whose sizes are in the range specified:

<table>
<thead>
<tr>
<th>Element</th>
<th>Packet Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Packet size not known</td>
</tr>
<tr>
<td>1</td>
<td>Reserved</td>
</tr>
<tr>
<td>2</td>
<td>Reserved</td>
</tr>
<tr>
<td>3</td>
<td>Reserved</td>
</tr>
<tr>
<td>4</td>
<td>0-15</td>
</tr>
<tr>
<td>5</td>
<td>16-31</td>
</tr>
<tr>
<td>6</td>
<td>32-63</td>
</tr>
<tr>
<td>7</td>
<td>64-127</td>
</tr>
<tr>
<td>8</td>
<td>128-255</td>
</tr>
<tr>
<td>9</td>
<td>256-511</td>
</tr>
<tr>
<td>10</td>
<td>512-1023</td>
</tr>
<tr>
<td>11</td>
<td>1024-2047</td>
</tr>
<tr>
<td>12</td>
<td>2048-4095</td>
</tr>
<tr>
<td>13-16</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

cb_msg_struct Structure for X.25

Purpose
Used by the x25_receive and x25_call_clear subroutines to pass the contents of a received packet to an application.

Syntax

```c
struct cb_msg_struct
{
    int msg_type;
    union
    {
        struct cb_call_struct *cb_call;
        struct cb_data_struct *cb_data;
        struct cb_clear_struct *cb_clear;
        struct cb_res_struct *cb_res;
        struct cb_int_struct *int_data;
    } msg_point;
};
```
**Fields**

msg_type  
Type of message being returned, as follows:

**X25_CALL_CONNECTED**  
Call connected: The cb_call field points to the cb_call_struct structure.

**X25_INCOMING_CALL**  
Incoming call: The cb_call field points to the cb_call_struct structure.

**X25_DATA**  
Data: The cb_data field points to the cb_data_struct structure.

**X25_DATA_ACK**  
Data acknowledgment: no buffer.

**X25_INTERRUPT**  
Interrupt: The int_data field points to the cb_int_data_struct structure.

**X25_INTERRUPT_CONFIRMATION**  
Confirmation of a previously issued interrupt request: no data is returned.

**X25_CLEAR_INDICATION**  
Indication that call has been cleared.

**X25_CLEAR_CONFIRM**  
Confirmation that the call has been cleared. The cb_clear field points to the cb_clear_struct structure. (This should only be received on a x25_call_clear call.)

**X25_RESET_INDICATION**  
Reset indication: The cb_res field points to the cb_res_struct structure.

**X25_RESET_CONFIRM**  
Reset confirmation: no data is returned.

**X25_UNKNOWN_PACKET**  
Allow for packets in future CCITT releases. These packets can be safely ignored by the application.

- **cb_call**  
  Pointer to the call structure, cb_call_struct

- **cb_data**  
  Pointer to the data structure, cb_data_struct

- **cb_clear**  
  Pointer to the clear structure, cb_clear_struct

- **cb_res**  
  Pointer to the reset structure, cb_res_struct

- **int_data**  
  Pointer to the interrupt data structure, cb_int_data_struct

---

**cb_pvc_alloc_struct Structure for X.25**

**Purpose**

Used by the x25_pvc_alloc subroutine to pass the name of the X.25 port and the logical channel number.

**Syntax**

```c
#define X25FLG_LINK_NAME 0x00000002
#define X25FLG_LCN 0x00000040

struct cb_pvc_alloc_struct
{
    unsigned long flags;
    char *link_name;
    unsigned int lcn;
};
```
Flags

X25_FLG_LCN Indicates that the lcn field is used.
X25_FLG_LINK_NAME Indicates that the link_name field is used.

Fields

flags Notification to the API that the associated field has been used.
link_name The name of the X.25 port.
lcn Logical channel number of the permanent virtual circuit (PVC) to be allocated to the call.

cb_res_struct Structure for X.25

Purpose
Used by the x25_reset and x25_receive subroutines to pass the reset cause and diagnostic codes.

Syntax
#define X25FLG_CAUSE 0x00000001
#define X25FLG_DIAGNOSTIC 0x00000002

struct cb_res_struct
{
    unsigned long flags;
    unsigned char cause;
    unsigned char diagnostic;
};

Flags

X25_FLG_CAUSE Indicates that the cause field is used.
X25_FLG_DIAGNOSTIC Indicates that the diagnostic field is used.

Fields

Structure field definitions are as follows:

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>flags</td>
<td>Notification to the API that the associated field has been used.</td>
</tr>
<tr>
<td>cause</td>
<td>Cause value of either 0 or in the range 0x80-0xFF, to be inserted in the reset packet.</td>
</tr>
<tr>
<td>diagnostic</td>
<td>Diagnostic reason to be inserted in the packet. The CCITT default value is 0.</td>
</tr>
</tbody>
</table>

ctr_array_struct Structure for X.25

Purpose
Used by the x25_ctr_wait subroutine to pass the counter identifier and a value to be exceeded.

Syntax
#define X25FLG_CTR_ID 0x00000001
#define X25FLG_CTR_VALUE 0x00000002

struct ctr_array_struct
{ unsigned long flags;
    int ctr_id;
    int ctr_value;
};

Flags

X25_FLG_CTR_ID Indicates that the ctr_id field is used.
X25_FLG_CTR_VALUE Indicates that the ctr_value field is used.

Fields

flags Notification to the API that the associated field has been used.
ctr_id Counter identifier.
ctr_value Value to be exceeded by the counter specified by the counter identifier. The counter is incremented each time a message for the associated call or PVC arrives. When the number of messages exceeds the value, the x25_ctr_wait subroutine returns control to the calling program.
Chapter 5. Directories

Directories contain directory entries. Each entry contains a file or subdirectory name and an i-node (index node reference) number. To increase speed and enhance the use of disk space, the data in a file is stored at various locations throughout the computer’s memory. The i-node contains the addresses used to locate all of the scattered blocks of data associated with a file. The i-node also records other information about the file, including time of modification and access, access modes, number of links, file owner, and file type. It is possible to link several names for a file to the same i-node by creating directory entries with the \texttt{ln} command.

Because directories often contain information that should not be available to all users of the system, directory access can be protected. See File ownership and user groups in Operating system and device management for more information.

Understanding Types of Directories

Directories can be defined by the system or the system administrator, or you can define your own directories. The system-defined directories contain specific kinds of system files, such as commands. At the top of the file system hierarchy is the system-defined root directory. The root directory is represented by a / (slash) and usually contains the following standard system-related directories:

- **/bin**: Symbolic link to the \texttt{/usr/bin} directory. In prior UNIX file systems, the \texttt{/bin} directory contained user commands that now reside in \texttt{/usr/bin} in the new file structure.
- **/dev**: Contains device nodes for special files for local devices. The \texttt{/dev} directory contains special files for tape drives, printers, disk partitions, and terminals.
- **/etc**: Contains configuration files that vary for each machine. Examples include:
  - \texttt{/etc/hosts}
  - \texttt{/etc/passwd}

  The \texttt{/etc} directory contains the files generally used in system administration. Most of the commands that used to reside in the \texttt{/etc} directory now reside in the \texttt{/usr/sbin} directory. However, for compatibility, it contains symbolic links to the new locations of some executable files. Examples include:
  - \texttt{/etc/chown} is a symbolic link to the \texttt{/usr/bin/chown}.
  - \texttt{/etc/exportvg} is a symbolic link to the \texttt{/usr/sbin/exportvg}.
- **/export**: Contains the directories and files on a server that are for remote clients.
- **/home**: Serves as a mount point for a file system containing user home directories. The \texttt{/home} file system contains per-user files and directories.

  In a standalone machine, a separate local file system is mounted over the \texttt{/home} directory. In a network, a server might contain user files that should be accessible from several machines. In this case, the server’s copy of the \texttt{/home} directory is remotely mounted onto a local \texttt{/home} file system.
- **/lib**: Symbolic link to the \texttt{/usr/lib} directory, which contains architecture-independent libraries with names in the form \texttt{lib*.a}.
- **/proc/sys**: Files in \texttt{/proc/sys} are used internally for kernel tuning and statistics gathering.
- **/sbin**: Contains files needed to boot the machine and mount the \texttt{/usr} file system. Most of the commands used during booting come from the boot image’s RAM disk file system; therefore, very few commands reside in the \texttt{/sbin} directory.
- **/tmp**: Serves as a mount point for a file system that contains system-generated temporary files.
- **/u**: Symbolic link to the \texttt{/home} directory.
- **/usr**: Serves as a mount point for a file system containing files that do not change and can be shared by machines (such as executables and ASCII documentation).

  Standalone machines mount a separate local file system over the \texttt{/usr} directory. Diskless and disk-poor machines mount a directory from a remote server over the \texttt{/usr} file system.
/var

Serves as a mount point for files that vary on each machine. The /var file system is configured as a file system since the files it contains tend to grow. For example, it is a symbolic link to the /usr/tmp directory, which contains temporary work files.

Some directories, such as your login or home directory ($HOME), are defined and customized by the system administrator. When you log in to the operating system, the login directory is the current directory. If you change directories using the cd command without specifying a directory name, the login directory becomes the current directory.

Related Information

Files, Directories, and File Systems for Programmers in AIX 5L Version 5.3 General Programming Concepts: Writing and Debugging Programs introduces i-nodes, file space allocation, and file, directory, and file system subroutines.

Directories in Operating system and device management introduces files and directories and the commands that control them.

/etc/locks Directory

Purpose

Contains lock files that prevent multiple uses of communications devices and multiple calls to remote systems.

Description

The /etc/locks directory contains files that lock communications devices and remote systems so that another user cannot access them when they are already in use. Other programs check the /etc/locks directory for lock files before attempting to use a particular device or call a specific system.

A lock file is a file placed in the /etc/locks directory when a program uses a communications device or contacts a remote system. The file contains the process ID number (PID) of the process that creates it.

The Basic Networking Utilities (BNU) program and other communications programs create a device lock file whenever a connection to a remote system, established over the specified device, is actually in use. The full path name of a device lock file is:

/etc/locks/DeviceName

where the DeviceName extension is the name of a device, such as tty3.

When the BNU uucico daemon, cu command, or tip command places a call to a remote system, it puts a system lock file in the /etc/locks directory. The full path name of a system lock file is:

/etc/locks/SystemName

where the SystemName extension is the name of a remote system, such as hera. The system lock file prevents more than one connection at a time to the same remote system.

Under normal circumstances, the communications software automatically removes the lock file when the user or program ends the connection to a remote system. However, if a process executing on the specified device or system does not complete its run (for example, if the computer crashes), the lock file remains in the /etc/locks directory either until the file is removed manually or until the system is restarted after a shutdown.
Related Information

The `connect` subcommand for the ATE command, `ct` command, `cu` command, `pdelay` command, `pshare` command, `slattach` command, `tip` command.

The `uucico` daemon.

**/usr/lib/hcon Directory**

**Purpose**
Contains files used by the Host Connection Program (HCON).

**Description**
The `/usr/lib/hcon` directory contains files used by the Host Connection Program (HCON). It contains color and keyboard definition files, terminal definition files, HCON API subdirectories, AUTOLOG example scripts, configuration data base files, and the command to start the HCON subsystem.

**Color and Keyboard Definition Files**
The following files contain data used to define and customize the HCON color and keyboard definition tables:

<table>
<thead>
<tr>
<th>File</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>e789_ctbl</td>
<td>Default binary color-definition table</td>
</tr>
<tr>
<td>e789_ktbl</td>
<td>Default binary keyboard-definition table</td>
</tr>
</tbody>
</table>

The color and keyboard definition tables in the `/usr/lib/hcon` directory specify defaults for use by HCON emulator sessions. The `hconutil` command allows users to customize color and keyboard definition tables.

**Terminal Definition Files**
The HCON installation process creates a `terminfo` subdirectory in the `/usr/lib/hcon` directory. The `/usr/lib/hcon/terminfo` directory contains terminal definition files that are specific to HCON. When HCON is installed, the `terminfo` directory contains the following files:

<table>
<thead>
<tr>
<th>File</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>ibm.ti.H</td>
<td>Terminal definitions for LFT, 5081, 3151, 3161, 3162, 3163, and 3164 terminals.</td>
</tr>
<tr>
<td>dec.ti.H</td>
<td>Terminal definitions for DEC VT100 and DEC VT220 terminals.</td>
</tr>
<tr>
<td>wyse.ti.H</td>
<td>Terminal definition for the WYSE WY-50 and WYSE WY-60 terminals.</td>
</tr>
</tbody>
</table>

The `terminfo` binary files for HCON terminal definitions are in subdirectories of the `/usr/lib/hcon/terminfo` directory. Each subdirectory is named with the first letter of the terminal name. When HCON is installed, the `terminfo` directory contains the following subdirectories:

<table>
<thead>
<tr>
<th>Subdirectory</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>Binary terminal definition file for running within the operating system windows</td>
</tr>
<tr>
<td>h</td>
<td>Binary terminal definition files for color and monochrome LFT</td>
</tr>
<tr>
<td>i</td>
<td>Binary terminal definition files for the 5081, 3151, 3161, 3162, and 3163 terminals</td>
</tr>
<tr>
<td>j</td>
<td>Binary terminal definition file for use with operating system windows</td>
</tr>
<tr>
<td>v</td>
<td>Binary terminal definition files for the DEC VT100 and DEC VT220 terminals</td>
</tr>
<tr>
<td>w</td>
<td>Binary terminal definition files for the WYSE WY-50 and WYSE WY-60 terminals</td>
</tr>
</tbody>
</table>

In addition to those delivered with HCON, the `/usr/lib/hcon/terminfo` subdirectory can contain customized terminal definitions.
**HCON API Subdirectories**
The HCON installation process creates two subdirectories in the `/usr/lib/hcon` directory that contain files used by the HCON API:

<table>
<thead>
<tr>
<th>Directory</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>mvs</td>
<td>API programs to use in interfacing to MVS/TSO host systems, including the <code>instalapi</code> program</td>
</tr>
<tr>
<td>vm</td>
<td>API programs to use in interfacing to VM/CMS host systems, including the <code>instalapi</code> program</td>
</tr>
</tbody>
</table>

**AUTOLOG Example Scripts**
The `/usr/lib/hcon` directory contains several example files for the AUTOLOG facility. These files are:

<table>
<thead>
<tr>
<th>File</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>logform</td>
<td>Example <code>genprof</code> form for creating AUTOLOG procedures</td>
</tr>
<tr>
<td>SYStso</td>
<td>Example AUTOLOG script for MVS/TSO host</td>
</tr>
<tr>
<td>SYSvm1</td>
<td>Example AUTOLOG script for VM/CMS host</td>
</tr>
<tr>
<td>SYSvm2</td>
<td>Example AUTOLOG script for VM/CMS host</td>
</tr>
</tbody>
</table>

**Configuration Data Base Files**
The following files contain HCON configuration information. This information is used by HCON programs, by the Object Data Manager (ODM), and by the HCON configuration commands, which are called by the System Management Interface Tool (SMIT).

<table>
<thead>
<tr>
<th>File</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysdflts</td>
<td>HCON database system defaults</td>
</tr>
<tr>
<td>sysdflts.vc</td>
<td>HCON database system defaults</td>
</tr>
<tr>
<td>users</td>
<td>HCON users database</td>
</tr>
</tbody>
</table>

**Command to Start the HCON Subsystem**
The `stcondmn` command is used to start the `hcondmn` subsystem after HCON has been installed.

**Files**

- `/usr/lib/hcon/terminfo` directory Contains terminal definitions.

**/var/spool/mqueue Directory for Mail**

**Purpose**
Contains the `log` file and temporary files associated with the messages in the mail queue.

**Description**
The `/var/spool/mqueue` directory contains temporary files associated with the messages in the mail queue and may contain the `log` file. For further information, see the `syslogd` daemon.

Temporary files have names that include the mail queue ID (`MQID`) of the message for which the file was created:

- `dfMQID` Data file
- `lfMQID` Lock file
- `ntMQID` Backup file
- `qtMQID` Queue control file
- `tfMQID` Temporary control file
- `xtMQID` Transcript file for session
**Related Information**

The `sendmail` command.

The `syslogd` daemon.

Mail queue concepts and tasks in *Networks and communication management*.

---

**/var/spool/uucp Directory for BNU**

**Purpose**
Stores Basic Networking Utilities (BNU) log, administrative, command, data, and execute files in multiple subdirectories.

**Description**
The `/var/spool/uucp` directory, also known as the BNU spooling directory, is the parent directory for multiple work directories created by the Basic Networking Utilities (BNU) program to facilitate file transfers among systems.

The following directories are subdirectories of the `/var/spool/uucp` directory:

- **.Admin**
  - Contains four administrative files, including:
    - `audit`
    - `Foreign`
    - `errors`
    - `xferstats`

- **.Corrupt**
  - Contains copies of files that could not be processed by the BNU program.

- **.Log**
  - Contains log files for the `uucico` and `uuxqt` daemons.

- **.Old**
  - Contains old log files for the `uucico` and `uuxqt` daemons.

- **.Status**
  - Records the last time the `uucico` daemon tried to contact remote systems.

- **.Workspace**
  - Holds temporary files that the file transport programs use internally.

- **.Xqtdir**
  - Contains execute files with lists of commands that remote systems can run.

- **SystemName**
  - Contains files used by file transport programs, including:
    - Command (C.*)
    - Data (D.*)
    - Execute (X.*)
    - Temporary (TM.*)

---

**Related Information**
The `uuclean` command, `uucp` command, `uudemon_cleanu` command, `uupick` command, `uuc` command, `uuto` command, `uux` command.

The `uucico` daemon, `uuxqt` daemon.

Understanding the BNU File and Directory Structure in *Networks and communication management*.
/var/spool/uucp/.Admin Directory for BNU

**Purpose**
Contains administrative files used by BNU.

**Description**
The `/var/spool/uucp/.Admin` directory contains administrative files used by the Basic Networking Utilities (BNU) program to facilitate remote communications among systems. The `.Admin` directory contains the following files:

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>audit</code></td>
<td>Contains debug messages from the <code>uucico</code> daemon.</td>
</tr>
<tr>
<td><code>Foreign</code></td>
<td>Logs contact attempts from unknown remote systems.</td>
</tr>
<tr>
<td><code>errors</code></td>
<td>Records <code>uucico</code> daemon errors.</td>
</tr>
<tr>
<td><code>xferstats</code></td>
<td>Records the status of file transfers.</td>
</tr>
</tbody>
</table>

**Related Information**
The `uudemon.cleanu` command.
The `cron` daemon, `uucico` daemon.

Understanding the BNU File and Directory Structure, BNU maintenance in *Networks and communication management*.

---

/var/spool/uucp/.Corrupt Directory for BNU

**Purpose**
Contains copies of files that could not be processed.

**Description**
The `/var/spool/uucp/.Corrupt` directory contains copies of files that could not be processed by the Basic Networking Utilities (BNU) program. For example, if a file is not in the correct form for transfer, the BNU program places a copy of that file in the `.Corrupt` directory for later handling. This directory is rarely used.

The files in the `.Corrupt` directory are removed periodically by the `uudemon.cleanu` command, a shell procedure.

**Related Information**
The `uudemon.cleanu` command.
The `uucico` daemon, `uuxqt` daemon.

Understanding the BNU File and Directory Structure, BNU maintenance in *Networks and communication management*.

---

/var/spool/uucp/.Log Directories for BNU

**Purpose**
Contain the BNU program log files.
Description

The `/var/spool/uucp/.Log` directories contain Basic Networking Utilities (BNU) program log files. The BNU program normally places status information about each transaction in the appropriate log file each time you use the networking utilities facility.

All transactions of the `uucico` and `uuxqt` daemons as well as the `uux` and `uucp` commands are logged in files named for the remote system concerned. Each file is stored in a subdirectory of the `/var/spool/uucp/.Log` directory, named for the daemon or command involved. Each subdirectory contains a separate file for each remote system contacted. Thus, the log files are named according to one of the following formats:

```
/var/spool/uucp/.Log/DaemonName/SystemName
```

OR

```
/var/spool/uucp/.Log/CommandName/SystemName
```

All activities of the `uucp` command are logged in the `SystemName` file in the `/var/spool/uucp/.Log/uucp` directory. All activities of the `uux` command are logged in the `SystemName` file in the `/var/spool/uucp/.Log/uux` directory.

The `uucp` and `uuto` commands call the `uucico` daemon. The `uucico` daemon activities for a particular remote system are logged in the `SystemName` file in the `/var/spool/uucp/.Log/uucico` directory on the local system.

The `uux` command calls the `uuxqt` daemon. The `uuxqt` daemon activities for a particular remote system are logged in the `SystemName` file in the `/var/spool/uucp/.Log/uuxqt` directory on the local system.

When more than one BNU process is running, however, the system cannot access the standard log file, so it places the status information in a file with a `.Log` prefix. The file covers that single transaction.

The BNU program can automatically append the temporary log files to a primary log file. This is called compacting the log files and is handled by the `uudemon.cleanu` command, a shell procedure. The procedure combines the log files of the activities of the `uucico` and `uuxqt` daemons on a particular system and stores the files in the `/var/spool/uucp/.Old` directory.

The default is for the `uudemon.cleanu` command to save log files that are two days old. This default can be changed by modifying the appropriate line in the shell procedure. If storage space is a problem on a particular system, reduce the number of days that the files are kept in their individual log files.

The `uulog` command can be used to view the BNU program log files.

Related Information

The `uucp` command, `uudemon.cleanu` command, `uulog` command, `uuto` command, `uux` command.

The `cron` daemon, `uucico` daemon, `uusched` daemon, `uuxqt` daemon.

[BNU log files] in *Networks and communication management*.

Understanding the BNU File and Directory Structure in *Networks and communication management*. 
/var/spool/uucp/.Old Directory for BNU

Purpose
Contains the combined BNU program log files.

Description
The /var/spool/uucp/.Old directory contains the combined Basic Networking Utilities (BNU) program log files.

The BNU program creates log files of the activities of the uucico and uuxqt daemons in the /var/spool/uucp/.Log directory. The log files are compacted by the /usr/sbin/uucp/uudemon.cleanu command, a shell procedure, which combines the files and stores them in the .Old directory.

By default, the uudemon.cleanu command removes log files after two weeks. The length of time log files are kept can be changed to suit the needs of an individual system.

The log files can be viewed using the uulog command.

Files
/var/spool/uucp/.Log directory Contains BNU program log files.

Related Information
The uucp command, uudemon.cleanu command, uulog command, uux command.

The cron daemon, uucico daemon, uuxqt daemon.

BNU log files in Networks and communication management.

Understanding the BNU File and Directory Structure in Networks and communication management.

/var/spool/uucp/.Status Directory for BNU

Purpose
Contains information about the status of the BNU program contacts with remote systems.

Description
The /var/spool/uucp/.Status directory contains information about the status of the Basic Networking Utilities (BNU) program contacts with remote systems.

For each remote system contacted, the BNU program creates a file in the .Status directory called SystemName, which is named for the remote system being contacted. In the .Status/SystemName file, the BNU program stores:
- Time of the last call in seconds
- Status of the last call
- Number of retries
- Retry time, in seconds, of the next call

**Note:** The times given in the .Status/SystemName file are expressed as seconds elapsed since midnight of January 1, 1970 (the output of a time subroutine). Thus, the retry time is in the form
of the number of seconds that must have expired since midnight of January 1, 1970, before the system can retry. To make this entry in the .Status/SystemName file, BNU performs a time subroutine, adds 600 seconds, and places the resulting number of seconds in the file.

If the last call was unsuccessful, the uucico daemon will wait until the time specified by the retry time before attempting to contact the system again. The retry time in the .Status/SystemName file can be overridden using the -r flag of the uutry or Uutry command.

Related Information

The uutry command, Uutry command.
The uucico daemon.
The time subroutine.

Understanding the BNU File and Directory Structure, Understanding the BNU Daemons in Networks and communication management.

/var/spool/uucp/SystemName Directories for BNU

Purpose
Contain queued requests for file transfers and command executions on remote systems.

Description
The /var/spool/uucp/SystemName directories are the Basic Networking Utilities (BNU) spooling directories on the local system. The BNU program creates a SystemName directory for each system listed in the /etc/uucp/Systems file, including the local system.

Each SystemName directory contains queued requests issued by local users for file transfers to remote systems and for command executions on remote systems.

The BNU program uses several types of administrative files to transfer data between systems. The files are stored in the SystemName directories:

- command (C.*) Contain directions for the uucico daemon concerning file transfers.
- data (D.*) Contain data to be sent to remote systems by the uucico daemon.
- execute (X.*) Contain instructions for running commands on remote systems.
- temporary (TM.*) Contain data files after their transfer to the remote system until the BNU program can deliver them to their final destinations (usually the /var/spool/uucppublic directory).

Related Information

The uucp command, uux command.
The uucico daemon, uusched daemon, uuxqt daemon.

Understanding the BNU Daemons, Understanding the BNU File and Directory Structure and BNU maintenance commands in Networks and communication management.

/var/spool/uucp/.Workspace Directory for BNU

Purpose
Holds temporary files used internally by file transport programs.
Description
The `/var/spool/uucp/.Workspace` directory holds temporary files of various kinds used internally by BNU file transport programs.

Related Information
The `uucico` daemon, `uuxqt` daemon.

Understanding the BNU File and Directory Structure in *Networks and communication management*.

/var/spool/uucp/.Xqtdir Directory for BNU

Purpose
Contains temporary files used by the `uuxqt` daemon to execute remote command requests.

Description
The `/var/spool/uucp/.Xqtdir` directory contains temporary files used by the Basic Networking Utilities (BNU) `uuxqt` daemon to execute remote command requests.

Related Information
The `uux` command.

The `uuxqt` daemon.

Understanding the BNU File and Directory Structure in *Networks and communication management*.

/var/spool/uucpppublic Directory for BNU

Purpose
Stores BNU files until they can be delivered.

Description
The `/var/spool/uucpppublic` directory is the public directory for the Basic Networking Utilities (BNU) facility. One of these directories exists on every system connected by the BNU utilities.

When a user transfers a file to a remote system or issues a request to execute a command on another system, the files generated by these BNU commands are stored in the public directory on the designated system until the destination directory is ready to receive them. (A user can also specify a destination other than the public directory when issuing the `uucp`, `uuto`, or `uux` command.) The transferred files remain in the `uucpppublic` directory until they are removed manually or automatically.

Note: The files are stored in the `uucpppublic/SystemName` subdirectory of the `uucpppublic` directory, where the `SystemName` directory is named for the remote system where the files originated.

All spooling directories are dynamic, including the public directory. Depending on the size of your installation and the number of files sent to the local `/var/spool/uucpppublic` directory by users on remote systems, this directory can become quite large.

The `uudemon.cleanu` command, a shell procedure, cleans up all the BNU spooling directories, including the public directories. Use the `uucleanup` command with the `-s SystemName` flag to clean up the directories on a specific system.
Related Information

The `uuclean` command, `uucleanup` command, `uucp` command, `uudemon.cleanu` command, `uuto` command, `uux` command.

Understanding the BNU File and Directory Structure in *Networks and communication management*. 
Appendix. Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user’s responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

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

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

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

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

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

IBM Corporation
Dept. LRAS/Bldg. 003
11400 Burnet Road
Austin, TX 78758-3498
U.S.A.

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

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:
IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM’s application programming interfaces.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:
(c) (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. (c) Copyright IBM Corp. _enter the year or years_. All rights reserved.

**Trademarks**

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

- AFS
- AIX
- AIX 5L
- IBM
- Lotus Notes
- MVS
- OS/390
- POWER3
- POWER4
- POWER5
Java and all Java-based trademarks and logos are registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.
Index

Special characters
/dev/error special file 766
/dev/errorctl special file 766
/dev/hty file 42
/dev/hyp file 42
/etc/group file 90
/etc/hty_config file format 432
/etc/locks directory 958
/etc/mail/aliases file (mail) 5
/etc/map3270 file format 470
/etc/mrouted.conf file 147
/etc/passwd file 176
/etc/rc.ntx file format 552
/etc/security/ids 99
/etc/security/audit/config file 22
/etc/security/audit/objects file 165
/etc/security/audit/streamcmds file 300
/etc/security/environ file 67
/etc/security/group file 91
/etc/security/lastlog file 435
/etc/security/limits file 179
/etc/security/roles file 223
/etc/security/smitacl.group file 273
/etc/security/smitacl.user file 275
/etc/security/sysck.cfg file 301
setting file definitions 301
/etc/security/user file 310
/etc/security/user.roles file 316
/etc/slp.conf file 271
/usr/lib/hcon directory 959
/usr/lib/security/audit/bincmds file 9
/usr/lib/security/audit/events file 75
/usr/lib/security/mkuser.default file 145
/usr/lib/sendmail.cf file (Mail) 226
/usr/spool/mqueue directory (Mail) 960
/usr/spool/uucp directory 961
/Admin directory 962
Audit file 6
errors file 73
Foreign file 87
xferstats file 327
/.Corrupt directory 962
/Status directory 964
/Workspace directory 965
/Xqtdir directory 966
SystemName directories 19, 41, 77, 306, 965
/usr/spool/uucp/public directory 966
/var/spool/uucp/public directory 962
.Log directories 962
.Old directory 964
.3270keys file format 333
.3270keys.hft file format 333
/Admin directory (BNU) 962
.fig file 85
.forward file 88
.ids file 99
.info file 101
.kshrc file 112
.Log directory (BNU) 962
.maildelivery file 131
.mailrc file
setting defaults for mail command 467
.mh_profile file 139
.netrc file format 529
.Old directory (BNU) 964
.aliases file format 560
.srf files
overview 300
.Status directory (BNU) 964
.times 306
.tiprc file format 641
.Workspace directory 965
.Xqtdir directory (BNU) 966
$HOME/.kshrc file 112

Numerics
32-bit file formats
ar_small 338
3270 connection adapter 752
3270 Host Connection Program/6000 959
400ap111845 314

A
a.out file format
standard common object file 685
accounting information
acct file format 335
acct.h file 335
accounting system
failed login file format 677
utmp file format 677
wtmp file format 677
acct file format 335
acct.cfg 2
acct.h file 335
administrative directory (BNU) 962
AIX V2 line discipline compatibility mode (termio.h file) 901
aliases
defining (MH) 476
definitions for sendmail command 5
file (mail) 5
ANY data type 669
API directories (HCON) 960
ar file format 336
arguments list structure
syntax 660
artwork
files
intermediate 85
ASCII
  file
    .fig 85
  asinfo file 61
Asynchronous Terminal Emulation 340
ATE
  ate.def file format 340
default file format 340
dialing directory file format 372
  phone numbers for remote connections 372
ate.def file format 340
attributes
  setting 90
    user 176
audit data structures 345
audit events 75
audit file 6
auditing system
  defining audit events 75
  defining auditstream commands 300
  defining commands to process bin files 9
  defining files for an audit 165
  defining the system configuration 22
audits
  defining the structure of accounting information files 924
  defining the structure of user information files 924
  describing auditing data structures 345
AUTOLOG example files (HCON) 960
auxiliary header (XCOFF) 689
auxiliary information
  symbol table (XCOFF) 717

B
backup file 7
  header records 7
Basic Networking Utilities 362
Berkeley line discipline (sgtty.h file) 879
bin files
  defining commands to process 9
bin stanza 23
binary
  file
    .srf 300
bincmds file 9
BNU
  /etc/locks directory 958
  /usr/spool/uucp directory 961
    .Admin directory 6, 73, 87, 327, 962
    .Corrupt directory 962
    .Status directory 964
    .Workspace directory 965
    .Xqtdir directory 966
    SystemName directories 19, 41, 77, 306, 965
  /usr/spool/uucppublic directory 966
  /var/spool/uucppublic directory
    .Log directories 962
    .Old directory 964
  administrative directory 962
  audit file 6
BNU (continued)
  command (C.*) files 19
  configuring 578
  cycling multispeed modems 584
  data (D.*) files 41
  data transferred from remote systems 306
  defining
    devices 362
    devices for remote communications 362
dialcodes 366
    modems and dialers 367
devices file format 362
dialcodes file format 366
dialers file format 367
ers errors file 73
execute (X.*) files 77
expect-send sequences 584
foreign file 87
limiting instances of
  ussched daemon 474
  uuxqt daemon 475
listing remote systems for communications 579
log access attempts by unknown systems 215
Maxuscheds file format 474
Maxuxqts file format 475
Permissions file format 535
poll file format 545
providing initial variable settings for the tip command 641
public directory 966
remote systems 41
remote systems for communications 367
remote.unknown file 215
specifying permissions for remote communications 535
specifying when to poll remote systems 545
spooling directory 961
systems file format 579
temporary (TM.*) files 306
tip command
  .tiprc file format 641
  phones file format 544
  remote file format 554
tunables file format 652
xferstats file 327
BNU file formats
  Sysfiles 578
boolean literal 669
BOOTP relay agent configuration file 10
bootparams file (NFS) 12
bootptab file format 347
buttons
  navigation
    labeling 112

C
C_BLOCK symbol
  XCOFF block auxiliary entry 727
C_EXT symbol
  XCOFF csect auxiliary entry 720
C_EXT symbol (continued)
  XCOFF function auxiliary entry 726
C_FCNS symbol
  XCOFF block auxiliary entry 727
C_FILE symbol
  XCOFF file auxiliary entry 718
C_HIDEXT symbol
  XCOFF csect auxiliary entry 720
  XCOFF function auxiliary entry 726
C_STAT
  section auxiliary entry 728
C_WEAKEXT symbol 720, 726
ca.cfg 12
callbacks list structure
  description 661
  syntax 661
cb_call_struct Structure for X.25 938
cb_circuit_info_struct Structure for X.25 939
cb_clear_struct Structure for X.25 940
cb_data_struct Structure for X.25 941
cb_dev_info_struct Structure for X.25 941
cb_fac_struct Structure for X.25 942
cb_int_data_struct Structure for X.25 947
cb_link_name_struct Structure for X.25 948
cb_link_stats_struct Structure for X.25 948
cb_msg_struct Structure for X.25 952
cb_pvc_alloc_struct Structure for X.25 953
cb_res_struct Structure for X.25 954
CD-ROM device driver 759
cdromd daemon 13
cdromd.conf 13
character set definition source file format
  CHARMAP section 348
  CHARSETID section 352
character set definitions
  CharacterSet keyword 681
  syntax 681
character set description 347
character sets
  parsing rules 666
charmap 347
CHARMAP section 348
CHARSETID section 352
child definitions
  Children keyword 684
  syntax 684
class definitions
  Class keyword 682
  control definition list 683
  modifications 683
  syntax 682
classes stanza 24
clsmnp.conf 16
code set
  code set converter
    define types for iconv character converters 835
code set maps
  file format 567
color definition table (HCON)
  default 65
  storing files 959
  combine file 336
command (C.*) files 19
compacted log files (BNU) 964
composite file header (XCOFF) 687
compound string literals 667
compver file 21
config file 22
configuration files
  BOOTP relay agent 10
  database (HCON) 960
  DHCP client 43
  DHCP server 46
  mrouted.conf 147
  NFS 181
  ntp.conf 155
  pam.conf 174
    changing 176
configuration information
  for gated daemon (TCP/IP) 383
  login authentication 120
  NIM 101
  user authentication 120
configure
  /etc/rc.ntx file 552
configuring for BNU 578
consdef file
  enabling asynchronous tty devices
    as console candidates 25
console special file 760
constants
  machine-defined 926
constraint arguments 660
control list definitions
  ControlList keyword 682
  syntax 682
control options
  file
    defining 831
controlling terminal interface
  supporting 812
controls list structure
  ChildName 661
  syntax 661
copyright file 27
copyright information file 27
core dump
  file format 353
core file format 357
crypto_file.h 822
crypto_has.pkf file 27
crypto_has.qkf file 29
crypto_has.thl file 31
cryptcasd.cfg file 32
ctr_array_struct Structure for X.25 954
clrmc.acls file 35
cltsec_map.global file 36
cltsec_map.local file 38
customizes the Korn-shell 112
customizing the MH package (MH) 139
cycling multispeed modems (BNU) 584
D

daemons (TCP/IP) 214
data (D.*) files 41
data storage consumption for string literals
  private, exported, and imported 668
data type definitions
  DataType keyword 680
  syntax 680
data types 665
  standard type definitions 922
  unsigned integers and addresses 923
databases
  defining locations of 109
dbx stabsstrings
  XCOFF section 736
debug section (XCOFF) 704
defines character symbols as character encodings 347
delta tables 562
describe the format of a compatible versions file 21
describing connections used by the tip command to
  contact remote systems (BNU) 544
describing remote systems contacted by the tip
  command (BNU) 554
description file
  legal lines of
    for troff command 649
description of management information base
  variables 478
determining default settings for ATE connections 340
device drivers
  error special files 766
devices
  preventing multiple uses of 958
devices file format 362
DHCP (dynamic host configuration protocol)
  BOOTP relay agent configuration file 10
  client configuration file 43
  server configuration file 46
DHCP client configuration file 43
DHCP server configuration file 46
dialcodes (BNU) 366
dialcodes file format 366
dialers file format 367
dialing directory file format 372
dials Special File 763
dir file 59, 192
directories
  description 957
dir file 59
  entries 957
  file system independent 824
  formats 59
  formatting entries 824
  HCON files 959
  i-node files 835
  naming 2
  types 957
diskette device-driver
  accessing 767
Display menu options
  specifying 112
difcn.h 825
difcn.h header file
  dynamic linking 824
DOMAIN files
  cache files
    file format 373
    standard resource record format 570
  data file
    format 374
    standard resource record 570
  local data files
    format 377
    standard resource record format 570
  reverse data files
    standard resource record format 570
dpid2.conf 60
dumpdates file 64
  holding date information for backup command 64
  holding date information for rdump command 64
DVD device driver
  mounting UDFS file system 14
Dynamic Host Configuration Protocol 43

E
e789_ctbl file 65
e789_ktbl file 65
eimadmin command 66
eimadmin.conf file 66
enumeration set definitions
  EnumerationSet keyword 682
  syntax 682
environ file 67
environment
  file 68
  setting by user 68
  variables
    defining 269, 270
    setting at login 546
eqn command
  special character definitions for 381
eqncchar file format 381
error image 357
error logging special files 766
errors file 73
Ethernet device handler
  accessing 765
ethers file for NIS 74
ethers files
  ethernet host addresses 74
euiciocl.h file 825
euiciocl.h files
  defining ioctl operations 825
exception section (XCOFF) 705, 707
executable file (XCOFF) 687
execute (X.*) files 77
expect-send sequences (BNU) 584
exports file (NFS) 79
expressions
  description 669
  set of operators in UIL 669
failed login file format 677
fig file 85
file definition
   setting 301
file formats 331
   acct 335
   bootptab 347
   core 353
   cpio 360
   failed login file format 677
   utmp file format 677
   widget meta-language 679
   WML 679
   wtmp file format 677
file header
   composite (XCOFF) 687
file mode interpretation
   using mode.h file 860
file system
   centralizing characteristics
      using filesystems file 85
   containing format of a logical volume
      using fs file 828
   copying into storage
      using backup file 7
   describing
      using inode file 835
   log attribute 85
   node name attribute 85
   size attribute 85
   type attribute 85
file systems
   /proc 192
file transfer (BNU)
   directions for the uucico daemon 19
   queued requests 965
files
   /etc/slp.conf 271
   .fig 85
   .srf 300
   archiving 338, 360
   backing-up 360
   ClientHostName.info 15
   containing text components 300
   control options
      defining 826, 831
   ct_ffdc.h 822
   ct_has.pkf 27
   ct_has.qkf 29
   ct_has.thl 31
   ctcasd.cfg 32
   crmc.acls 35
   ctsec_map.global 36
   ctsec_map.local 38
   data types
      defining primitive system 922
dir 59
dlfcn.h 825
environment 68
ethers 74
groups
  setting basic attributes  90
  setting extended attributes  91
grp.h file  834

H
hardware buses
  accessing  758
hardware flow control operations  921
hardware parameters  864
HCON
  /usr/lib/hcon directory  959
  API subdirectories  960
  AUTOLOG example files  960
  configuration database files  960
  directory  959
  e789_ctbl file  65
  e789_ktbl file  65
  files  959
  LAF example files  960
  storing color definition table files  959
  terminal definition files  959
header files  885
  acct.h  335
  control block
    list of  820
  fcntl.h  826
  flock.h  831
  fullstat.h  833
  ipc.h  840
  limits.h  843
  math.h  859
  mode.h  860
  msg.h  862
  param.h  864
  poll subroutine structures  871
  sem.h  877
  sgtty.h  879
  shm.h  885
  spc.h  886
  srcobj.h  890
  stat.h  891
  statfs.h  893
  statvfs.h  895
  termio.h  901
  termios.h  910
  termiox.h  920
  types.h  922
  unistd.h  923
  value.h  926
  vmount.h  927
  XCOFF  687
holding internal files for remote communications (BNU)  965
host names and addresses (TCP/IP)  426
host-adapter raw interface
  defining  42
  hostmibd.conf  94
  hosts file format  426
  hosts.equiv file format  428
  hosts.lpd file format  431
  hty configuration
    hty_config file format  432

I
i-nodes  957
i-numbers  957
iconv.h file
  defining iconv character converters  835
IDE adapter driver
  accessing  772
identifier sections
  identifier  663
  syntax  663
image.data file  95
  describing installed images  95
  include directives
    description  663
    syntax  663
indexed archives  338
lNed files
  programs and data used  100
inetd.conf file format
  service requests  433
  initialization process  103
  inittab file  103
  inode.h file  835
  integer literals
    data storage consumption  668
    description  668
  intermediate files
    .fig  85
    .srf (text)  300
  inttypes.h file  839
    fixed size integral types  839
  ipc.h File  840
  irs.conf file  105
  iso646.h file  840
    providing alternate spellings  840
  ispaths file  109
  isprime file  112
    creating for library  112
    overview  112

K
kbd Special File
  accessing natively attached keyboards  772
keyboard definition table (HCON)
  default  65
    storing files  959
Korn-shell
  customizes  112

L
labels
  for navigation buttons
    specifying  112
LAF (Login Assist Facility) example files (HCON)  960
language syntax 663
lastlog file format 435
LC_COLLATE category 441, 443
LCCTYPE category 441, 446
LC_MESSAGES category 441, 449
LC_MONETARY category 441, 450
LC_NUMERIC category 441, 454
LC_TIME category 441, 456
LDAP Attribute Mapping 440
ldap.cfg file format 436
ldapid.ldif.template file 116
ldr.h 841
lft special file 774
providing character-based terminal support
for local graphics displays and keyboards 773
libperfstat.h file 845
Libraries
libperfstat.h 845
library
location of databases in 109
limits file 117
line disciplines
AIX V2 compatibility mode (termio.h file) 901
Berkeley (sgtty.h file) 879
POSIX (termios.h file) 910
line number (XCOFF) 712
line printer device driver
accessing 775
links
to navigation articles
specifying 112
list
F files
.fig file 85
list groups of users 149
list package contents 186
list sections
arguments list structure 660
syntax 660
loadable authentication module configuration
information 134
loader section (XCOFF) 698
local loopback information for named (TCP/IP) 377
local user name 382
local_domain 119
locale definition source file format 441
LC_COLLATE category 441, 443
LCCTYPE category 441, 446
LC_MESSAGES category 441, 449
LC_MONETARY category 441, 450
LC_NUMERIC category 441, 454
LC_TIME category 441, 456
locale method source file format 460
lock files
storing devices and remote systems 958
log access attempts by unknown systems (BNU) 215
log files (BNU)
access attempts by unknown systems 87
compacted 964
directory 962
logical volume
containing format of a file system 828
logical volume device driver
accessing 779
login attempt information 191
login authentication
configuration information 120
login.cfg file 120
lp special file 775
lpfk special file 778
M
machine boot process 213
magic file
defining file types
/etc/magic file 466
mail
/etc/mail/sendmail.cf file
classes 242
macros 232, 233
message headings 247
rewrite rules 227
rule sets 227
automatically forwards 88
mail command default settings 467
Mail files
/etc/mail/aliases 5
/usr/lib/sendmail.cf 226
/usr/spool/mqueue directory 960
mail queue files directory 960
management information base variables 478
mapping
UCS-2 conversion 654
maps
public and secret keys 209
math constants
defined in the math.h file 859
math.h file 859
Maxuuscheds file format 474
Maxuxqts file format 475
menu options
Display
specifying 112
Message Handler 146
methods.cfg File 134
MH
.mh_profile file 139
mailedelivery file 131
mh_alias file format 476
mhl.format file 136
mtstailor file 146
mh_alias file format 476
mhl.format file 136
mb.defs file format 478
mibill file 143
mosy command 143
mkuser.default file 145
modem control operations 919
modems (BNU)
cycling multispeed modems 584
modems (BNU) \textit{(continued)}
\begin{itemize}
  \item expect-send sequences \textbf{584}
\end{itemize}
module
\begin{itemize}
  \item pam\_aix \textbf{166}
  \item pam\_allow \textbf{168}
  \item pam\_allowroot \textbf{169}
  \item pam\_ckfile \textbf{170}
  \item pam\_permission \textbf{171}
  \item pam\_prohibit \textbf{172}
  \item pam\_rhosts\_auth \textbf{173}
\end{itemize}
mouse special file \textbf{786}
mpcn special file \textbf{787}
\begin{itemize}
  \item error codes \textbf{788}
\end{itemize}
mpqi special file \textbf{789}
MPOP device handler
\begin{itemize}
  \item accessing \textbf{790}
  \item error conditions \textbf{790}
  \item system call support \textbf{790}
\end{itemize}
mqueue directory (Mail) \textbf{960}
mrouted.conf file \textbf{147}
msg.h File \textbf{862}
mtlo.h file \textbf{864}
mtstailor file \textbf{146}
multiple screen utility
\begin{itemize}
  \item terminal descriptions (asinfo file) \textbf{61}
\end{itemize}
MultiProtocol Quad Port device handler \textbf{790}

\textbf{N}
\begin{itemize}
  \item name resolution
    \begin{itemize}
      \item DOMAIN cache file format \textbf{373}
      \item DOMAIN data file format \textbf{374}
      \item DOMAIN local data file format \textbf{377}
      \item DOMAIN reverse data file format \textbf{379}
      \item named.conf file format \textbf{479}
      \item standard resource record format \textbf{570}
    \end{itemize}
  \item name resolution services \textbf{105}
  \item ordering \textbf{151}
\end{itemize}
name table
\begin{itemize}
  \item XCOFF loader import file ID \textbf{702}
\end{itemize}

named.conf file format \textbf{479}

names and strings
\begin{itemize}
  \item object types \textbf{664}
  \item reserved keywords \textbf{664}
\end{itemize}

navigation article
\begin{itemize}
  \item links to
    \begin{itemize}
      \item specifying labels for \textbf{112}
      \item primary
        \begin{itemize}
          \item specifying labels for links \textbf{112}
          \item specifying labels for buttons \textbf{112}
        \end{itemize}
    \end{itemize}
\end{itemize}

neqn command
\begin{itemize}
  \item special character definitions for \textbf{381}
\end{itemize}

netgroup file
\begin{itemize}
  \item /etc/netgroup
    \begin{itemize}
      \item network users list \textbf{149}
    \end{itemize}
\end{itemize}

netmasks file \textbf{150}
\begin{itemize}
  \item network masks
    \begin{itemize}
      \item implementing IP standard subnetting \textbf{150}
    \end{itemize}
\end{itemize}

netsvc.conf file \textbf{151}

netsvc.conf file format
\begin{itemize}
  \item specifying name resolution service order \textbf{151}
\end{itemize}

Network File System \textbf{966}

Network Information Service \textbf{966}

network masks \textbf{150}

Network Terminal Accelerator files \textbf{42, 432, 552}

networks file (NFS) \textbf{153}

networks file format \textbf{531}

NFS
\begin{itemize}
  \item local\_domain \textbf{119}
  \item realm-to-domain
    \begin{itemize}
      \item realm.map file \textbf{215}
      \item security\_default \textbf{226}
    \end{itemize}
\end{itemize}

NFS files
\begin{itemize}
  \item bootparams \textbf{12}
  \item exports \textbf{79}
  \item networks \textbf{153}
  \item pcnfsd configuration \textbf{181}
  \item rpc \textbf{224}
  \item xtab \textbf{328}
\end{itemize}

NIM
\begin{itemize}
  \item configuration information \textbf{101}
\end{itemize}

NIS
\begin{itemize}
  \item netmask \textbf{150}
\end{itemize}

NLSvec file
\begin{itemize}
  \item encoding PostScript fonts \textbf{153}
\end{itemize}

nonvolatile RAM
\begin{itemize}
  \item platform-specific
    \begin{itemize}
      \item accessing \textbf{792}
    \end{itemize}
\end{itemize}

nroff command
\begin{itemize}
  \item setting terminal driving tables \textbf{532}
\end{itemize}

nterm file format \textbf{532}

ntp.conf file \textbf{155}

ntp.keys file
\begin{itemize}
  \item authentication of NTP transactions
    \begin{itemize}
      \item key and key identifiers \textbf{164}
    \end{itemize}
\end{itemize}

null device
\begin{itemize}
  \item accessing \textbf{791}
  \item null special file \textbf{791}
\end{itemize}

\textbf{O}

object file (XCOFF) \textbf{687}

object file format \textbf{685}

object sections
\begin{itemize}
  \item description \textbf{662}
  \item forward-referenced \textbf{662}
\end{itemize}

ObjectType \textbf{662}

syntax \textbf{662}

TagValue \textbf{662}

objects file \textbf{165}

optical media device driver
\begin{itemize}
  \item accessing \textbf{794}
\end{itemize}

output format control for the mhl command (MH) \textbf{136}

\textbf{P}

package characteristics file
\begin{itemize}
  \item format \textbf{184}
\end{itemize}

pam\_aix \textbf{166}

pam\_allow \textbf{168}

pam\_allowroot \textbf{169}

pam\_ckfile \textbf{170}
resource definitions
   class definitions 685
   Resource keyword 684
   syntax 684
retry time
   before calling a remote system (BNU) 964
reverse data file format 379
rewrite rules
   mail 227
roles file 223
routed daemon
   gateways file format 424
rpc file (NFS) 224
rule sets
   mail 227

S
   sample input to mosy 143
   sample snmpd agent configuration 276
   sccsfile 562
   SCSI adapter driver
      accessing 807
   SCSI tmscsi device driver
      accessing 809
   secret key maps 209
   section headers (XCOFF) 693, 696
   sections (XCOFF) 696
   sectoldif.cfg 225
security
   .ids file 99
security (BNU)
   logging access attempts by unknown systems 87
   recording access attempts by unknown systems 215
   specifying permissions for remote communications 535
security_default 226
semaphore operations
   sem.h file 877
sendmail command
   alias definitions 5
sendmail configuration data 226
sendmail.cf file (Mail) 226
Serial Optical Link
   accessing 797
Serial Optical Link device driver
   opn special file 797
services file format 566
setmaps file format 567
sets up user environment 68
setting
   default gateway 551
   host ID 551
   host name 551
   static route 551
setting defaults for mail command 467
setup.csh file 269
setup.sh file 270
sgtty.h file 879
shm.h File 885
simprof file format 568
slp.conf 271
Small Computer System Interface 807
smi.my file
   sample SMI input
   mosy command 271
smitacl.group file
   group access control list definitions 273
smitacl.user file 275
snmpdv3.conf file 276
snmpmibd.conf 297
snmpt.boots
   machine boot process 284
Source Code Control System (SCCS) 562
special files 760
   3270 connection adapter 752
   accessing tablet file 807
   CD-ROM device driver 759
   cdromd.conf 13
   controlling terminal interfaces with dials 763
   diskette device-driver 767
   error 766
   errorctl 766
   Ethernet adapter 765
   hardware buses 758
   IDE adapter driver 772
   line printer device driver 775
   logical volume device driver 779
   lp 775
   lpfk 778
   mouse 786
   mpcn 787
   mpqi 789
   mpqn 790
   nonvolatile RAM 792
   null device 791
   opn 797
   optical media device driver 794
   physical volumes device driver 801
   privileged virtual memory
      read access 783
      write access 783
   SCSI adapter driver 807
   SCSI tmscsi device driver 809
   sequential-access bulk storage medium device driver 803
   Serial Optical Link 797
   system dump 764
   token-ring device handler 809
   X.25 co-processor/2 adapter 815
   spooling directory (BNU) 961
SRC
   SRC process structures 890
   subsystem process structures 886
SRC subsystem programming requirements
   SRC request packets
   SRC request structure example 889
srf file 300
stabstrings section (XCOFF) 736
standard resource record format 573
address records 573
canonical name records 574
domain name pointer records 575
gateway ptr records 575
host information records 573
IN-ADDR.ARPA record 575
mail exchanger records 577
mail group member records 577
mail rename name records 576
mailbox information records 577
mailbox records 576
name server records 573
start of authority records 572
well-known services records 574
standards
environment implementation 923
implementation limits
ANSI C 843
IEEE P1003 POSIX 843
start-up file format 568
statistics
returning file 893
statistics about status of file transfer requests (BNU) 327
statistics subroutines
structuring of returned data 893
status of calls to remote systems (BNU) 964
status subroutines
header file structure 891
statvfs subroutine statistics
structure 895
storage
combined log files (BNU) 964
debugging messages form the uucico daemon (BNU) 6
files awaiting transfer 961
files that cannot be transferred (BNU) 962
lock files that prevent multiple uses of communication devices 958
log and administrative files (BNU) 961
reverse name resolution information for named (TCP/IP) 379
transferred files until delivered (BNU) 966
stream stanza 24
streamcmds file 300
string literals
escape sequences 667
supported character sets 666
syntax 665
string table 735
XCOFF 735
XCOFF loader section 703
symbol table 728
XCOFF 728
XCOFF loader section 699
symbol table (XCOFF) 713
sysck.cfg file 301
Sysfiles file format 578
syslog.conf file 303
system consoles
accessing 760
system files 1, 2
System Resource Controller 886
systemcfg.h file 895
defining _system_configuration structure 895
SystemName directories (BNU) 965
Systems file format 579

T
tablet special file 807
tailoring MH environments (MH) 146
tar.h file
/usr/include/tar.h 897
tar archive header 897		
targets 304
TCP/IP 566
BNU with
tables in the Dialers file 370
tables in the Systems file 584
file formats
/etc/map3270 470
.3270 keys 333
.netrc 529
.rhosts 560
DOMAIN cache 373
DOMAIN data 374
DOMAIN local data 377
DOMAIN reverse data 379
ftpusers 382
gated.conf 383
gateways 424
hosts 426
hosts.equiv 428
hosts.lpd 431
inetcfg 432
Internet services 566
netdir.conf 479
networks 531
protocols 547
rc.net 549
resolv.conf 557
files
mrouted.conf 147
rc.tcpip 214
remote hosts
specifying to print on a local host 431
standard resource record format 570
telnet.conf file (TCP/IP) 586
temporary (TM.*) files 306
temporary uuxqt daemon work files, directory for BNU 966
terminal capabilities
color manipulation 627
functions
area clears 618
basic 615
cursor motions 618
highlighting, underlining, and visual bells 621
insert or delete line character 620
terminal capabilities (continued)
functions (continued)
  insert or delete lines 619
  keypad 623
  parameterized strings 617
  scrolling 619
  general 588
  line graphics 626
  miscellaneous strings 625
  status lines 626
  tabs and initialization 624
types
  Boolean 589
  numeric 591
  string 593
terminal definition files (HCON) 959
terminal descriptions
  asinfo file 61
  terminfo database 587
terminal interface 879
  controlling 812
  pseudo terminal 799
  virtual terminal server 813
terminal map
  file format 567
terminfo
  database 587
  entry 639
  file format 587
  file names 639
  preparing descriptions 615
  printers and 629
  similar terminals and 629
  source file entries 587
  special cases 628
termio.h file 901
termios.h file 910
termiox.h file 920
threads list 872
.tip command
  .tiprc file format 641
  contacting remote systems 544, 554
  phones file format 544
  providing initial variable settings 641
  remote file format 554
tmscsci device driver
  accessing 809
token-ring adapter 809
Token-Ring device handler
  subroutine support 810
  using 810
trace special file
  event tracing 811
translating terminal_type strings 586
trcfmt file format
  storing trace templates 642
troff
  command
    specifying description files 649
    file format 646
    font file format 649
  command 646
troff file format 647
trusted computing base 301
TTY interface
  defining 42
TTY subsystem 587, 901
  controlling terminal 812
tunables file format 652
type-check section (XCOFF) 704
U
unconvdef Source File Format 654
UDFS
  mounting options 14
UIL file format
  compiler 657
  description 656
UNIX-to-UNIX Copy Program (UUCP) 362
unix.map file 308
updaters file
  updating NIS maps 306
urandom 793, 813
user
  environment
    setting at login 546
    file 310
  interface language file format 656
  keyboard mapping and colors
    telnet command 333
    tn3270 command 470
  setting
    basic attributes 176
    default attributes 145
    environment attributes 67
    extended attributes 310
    password attributes 179
  process resource hard limits 117
user ACL definitions list 275
user authentication
  configuration information 120
user.roles file 316
users stanza 24
utmp file format 677
uucico daemon
  debugging messages from 6
  file transfer directions
    files of 19
  limiting instances of 474
  log files 962
  recording errors from 73
UUCP 362
uucp command
  log files 962
uusched daemon
  limiting instances of 474
uxx command
  executing log files 962
uxxqt daemon
  executing log files 962
  limiting instances of 475
Index 985

V
value declaration 671
value sections Supported value types 658 syntax 658 ValueExpression 658 ValueName 658 ValueType 658 value selections 658 variables environment 270 versions file, describe the format 21 VFS data structure definitions 927 executing the vfs file 317 vfs file 317 vgrind command language definition database 677 vgrinddefs file format 677 Virtual File Systems 317 vts 813 vty_server 813

W
wctype.h file 928 wide-character classification list 928 widget meta-language 679 comments syntax 680 syntax elements 679 WLM .times file 306 groupings file format 93 wlm.h file 929 WML file format comments syntax 680 description 679 syntax elements 679 xtab file format 677

X
We appreciate your comments about this publication. Please comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.

For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state on this form.

Comments:

Thank you for your support.
Submit your comments using one of these channels:
• Send your comments to the address on the reverse side of this form.
• Send your comments via e-mail to: aix6koub@austin.ibm.com

If you would like a response from IBM, please fill in the following information:

<table>
<thead>
<tr>
<th>Name</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company or Organization</td>
<td></td>
</tr>
<tr>
<td>Phone No.</td>
<td>E-mail address</td>
</tr>
</tbody>
</table>