Bull

AIX 5L Security Guide

AIX

ORDER REFERENCE
86 A2 57EM 02

Bull

AIX 5L Security Guide

AIX

Software

October 2005

BULL CEDOC
357 AVENUE PATTON
B.P.20845
49008 ANGERS CEDEX 01
FRANCE

ORDER REFERENCE
86 A2 57EM 02

Suggestions and criticisms concerning the form, content, and presentation of this book are invited. A form is provided at the end of this book for this purpose.

To order additional copies of this book or other Bull Technical Publications, you are invited to use the Ordering Form also provided at the end of this book.

Trademarks and Acknowledgements

We acknowledge the right of proprietors of trademarks mentioned in this book.

AIX® is a registered trademark of International Business Machines Corporation, and is being used under licence.

UNIX is a registered trademark in the United States of America and other countries licensed exclusively through the Open Group.

Linux is a registered trademark of Linus Torvalds.

The information in this document is subject to change without notice. Bull will not be liable for errors contained herein, or for incidental or consequential damages in connection with the use of this material.

Contents

About this book

This book provides system administrators with complete information on file, system, and network security. This guide contains information about how to perform such tasks as hardening a system, changing permissions, setting up authentication methods, and configuring the Common Criteria Security Evaluation features. This publication is also available on the documentation CD that is shipped with the operating system.

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 publications contain related information:

- *AIX 5L Version 5.3 System Management Guide: Operating System and Devices*
- *AIX 5L Version 5.3 System Management Concepts: Operating System and Devices*
- *AIX 5L Version 5.3 System Management Guide: Communications and Networks*
- *AIX 5L Version 5.3 Installation Guide and Reference*
- *AIX 5L Version 5.3 Commands Reference*
- *AIX 5L Version 5.3 Files Reference*
- *AIX 5L Version 5.3 General Programming Concepts: Writing and Debugging Programs*
- *AIX 5L Version 5.3 System User's Guide: Operating System and Devices*
- *AIX 5L Version 5.3 System User's Guide: Communications and Networks*
- *AIX 5L Version 5.3 Network Information Services (NIS and NIS+) Guide*
- *AIX 5L Version 5.3 Guide to Printers and Printing*

Securing the Base Operating System

Securing the Base Operating System provides information about how to protect the system regardless of network connectivity.

These sections describe how to install your system with security options turned on, and how to secure AIX® against nonprivileged users gaining access to the system.

Secure system installation and configuration

Topics in this section include:

- "Trusted Computing Base"
- "Controlled Access Protection Profile and Evaluation Assurance Level 4+" on page 6
- "Login control" on page 17
- "Managing X11 and CDE concerns" on page 23

Trusted Computing Base

The system administrator must determine how much trust can be given to a particular program. This determination includes considering the value of the information resources on the system in deciding how much trust is required for a program to be installed with privilege.

The Trusted Computing Base (TCB) is the part of the system that is responsible for enforcing system-wide information security policies. By installing and using the TCB, you can define user access to the trusted communication path, which allows for secure communication between users and the TCB. TCB features can only be enabled when the operating system is installed. To install TCB on an already installed machine, you will have to perform a Preservation installation. Enabling TCB allows you to access the trusted shell, trusted processes, and the Secure Attention Key (SAK).

This section discusses the following topics:

- "Installing a system with the TCB"
- "Checking the TCB" on page 2
- "Structure of the sysck.cfg file" on page 2
- "Using the tcbck command" on page 3
- "Configuring additional trusted options" on page 5

Installing a system with the TCB

The TCB is the part of the system that is responsible for enforcing the information security policies of the system. All of the computer's hardware is included in the TCB, but a person administering the system should be concerned primarily with the software components of the TCB.

If you install a system with the Trusted Computing Base option, you enable the trusted path, trusted shell, and system-integrity checking (**tcbck** command). These features can *only* be enabled during a base operating system (BOS) installation. If the TCB option is not selected during the initial installation, the **tcbck** command is disabled. You can use this command only by reinstalling the system with the TCB option enabled.

To set the TCB option during a BOS installation, select **More Options** from the Installation and Settings screen. In the Installation Options screen, the default for the **Install Trusted Computing Base** selection is **no**. To enable the TCB, type 2 and press Enter.

Because every device is part of the TCB, every file in the **/dev** directory is monitored by the TCB. In addition, the TCB automatically monitors over 600 additional files, storing critical information about these files in the **/etc/security/sysck.cfg** file. If you are installing the TCB, immediately after installing, back up this file to removable media, such as tape, CD, or disk, and store the media in a secure place.

Checking the TCB

The security of the operating system is jeopardized when the Trusted Computing Base (TCB) files are not correctly protected or when configuration files have unsafe values.

The **tcbck** command audits the security state of the Trusted Computing Base. The **tcbck** command audits this information by reading the **/etc/security/sysck.cfg** file. This file includes a description of all TCB files, configuration files, and trusted commands.

The **/etc/security/sysck.cfg** file is not offline and, could therefore be altered by a hacker. Make sure you create an offline read-only copy after each TCB update. Also, copy this file from the archival media to disk before doing any checks.

Installing the TCB and using the **tcbck** command do not guarantee that a system is operating in a Controlled Access Protection Profile (CAPP) and Evaluation Assurance Level 4+ (EAL4+) compliant mode. For information on the CAPP/EAL4+ option, see “Controlled Access Protection Profile and Evaluation Assurance Level 4+” on page 6.

Structure of the sysck.cfg file

The **tcbck** command reads the **/etc/security/sysck.cfg** file to determine which files to check. Each trusted program on the system is described by a stanza in the **/etc/security/sysck.cfg** file.

Each stanza has the following attributes:

acl
Text string representing the access control list for the file. It must be of the same format as the output of the **aclget** command. If this does not match the actual file ACL (access control list), the **sysck** command applies this value using the **aclput** command.

Note: The SUID, SGID, and SVTX attributes must match those specified for the mode, if present.

class
Name of a group of files. This attribute allows several files with the same class name to be checked by specifying a single argument to the **tcbck** command. More than one class can be specified, with each class being separated by a comma.

group
Group ID or name of the file group. If this does not match the file owner, the **tcbck** command sets the owner ID of the file to this value.

links
Comma-separated list of path names linked to this file. If any path name in this list is not linked to the file, the **tcbck** command creates the link. If used without the *tree* parameter, the **tcbck**command prints a message that there are extra links but does not determine their names. If used with the *tree* parameter, the **tcbck** command also prints any additional path names linked to this file.

mode
Comma-separated list of values. The allowed values are SUID, SGID, SVTX, and TCB. The file permissions must be the last value and can be specified either as an octal value or as a 9-character string. For example, either 755 or rwxr-xr-x are valid file permissions. If this does not match the actual file mode, the **tcbck** command applies the correct value.

owner
User ID or name of the file owner. If this does not match the file owner, the **tcbck** command sets the owner ID of the file to this value.

program
Comma-separated list of values. The first value is the path name of a checking program. Additional values are passed as arguments to the program when it is executed.

Note: The first argument is always one of *-y*, *-n*, *-p*, or *-t*, depending on which flag the **tcbck** command was used with.

source Name of a file this source file is to be copied from prior to checking. If the value is blank, and this is either a regular file, directory, or a named pipe, a new empty version of this file is created if it does not already exist. For device files, a new special file is created for the same type device.

symlinks Comma-separated list of path names symbolically linked to this file. If any path name in this list is not a symbolic link to the file, the **tcbck** command creates the symbolic link. If used with the *tree* argument, the **tcbck** command also prints any additional path names that are symbolic links to this file.

If a stanza in the **/etc/security/sysck.cfg** file does not specify an attribute, the corresponding check is not performed.

Using the tcbck command

The **tckck** command is used to ensure the proper installation of security-relevant file; to ensure the file system tree contains no files that clearly violate system security; and to update, add, or delete trusted files.

The **tcbck** command is normally used for the following tasks:

- Ensure the proper installation of security-relevant files
- Ensure that the file system tree contains no files that clearly violate system security
- Update, add, or delete trusted files

The **tcbck** command can be used in the following ways:

- Normal use
  - Noninteractive at system initialization
  - With the **cron** command
- Interactive use
  - Check out individual files and classes of files
- Paranoid use
  - Store the **sysck.cfg** file offline and restore it periodically to check out the machine

Although not cryptographically secure, the TCB uses the **sum** command for checksums. The TCB database can be set up manually with a different checksum command, for example, the **md5sum** command that is shipped in the textutils RPM Package Manager package with *AIX Toolbox for Linux Applications CD*.

Checking trusted files:

Use the **tckck** command to check and fix all the files in the tckck database, and fix and produce a log of all errors.

To check all the files in the tcbck database, and fix and report all errors, type:

```
tcbck -y ALL
```

This causes the **tcbck** command to check the installation of each file in the tcbck database described by the **/etc/security/sysck.cfg** file.

To perform this automatically during system initialization, and produce a log of what was in error, add the previous command string to the **/etc/rc** command.

Checking the file system tree:

Whenever you suspect the integrity of the system might have been compromised, run the **tcbck** command to check the file system tree.

To check the file system tree, type:

```
tcbck -t tree
```

When the **tcbck** command is used with the *tree* value, all files on the system are checked for correct installation (this could take a long time). If the **tcbck** command discovers any files that are potential threats to system security, you can alter the suspected file to remove the offending attributes. In addition, the following checks are performed on all other files in the file system:

- If the file owner is root and the file has the SetUID bit set, the SetUID bit is cleared.
- If the file group is an administrative group, the file is executable, and the file has the SetGID bit set, the SetGID bit is cleared.
- If the file has the **tcb** attribute set, this attribute is cleared.
- If the file is a device (character or block special file), it is removed.
- If the file is an additional link to a path name described in **/etc/security/sysck.cfg** file, the link is removed.
- If the file is an additional symbolic link to a path name described in **/etc/security/sysck.cfg** file, the symbolic link is removed.

Note: All device entries must have been added to the **/etc/security/sysck.cfg** file prior to execution of the **tcbck** command or the system is rendered unusable. To add trusted devices to the **/etc/security/sysck.cfg** file, use the **-l** flag.

Attention: *Do not* run the **tcbck -y tree** command option. This option deletes and disables devices that are not properly listed in the TCB, and might disable your system.

Adding a trusted program:

Use the **tckck** command to add a specific program to the **/etc/security/sysck.cfg** file.

To add a specific program to the **/etc/security/sysck.cfg** file, type:

```
tcbck -a PathName [Attribute=Value]
```

Only attributes whose values are not deduced from the current state of the file need be specified on the command line. All attribute names are contained in the **/etc/security/sysck.cfg** file.

For example, the following command registers a new SetUID root program named **/usr/bin/setgroups**, which has a link named **/usr/bin/getgroups**:

```
tcbck -a /usr/bin/setgroups links=/usr/bin/getgroups
```

To add `jfh` and `jsl` as administrative users and to add `developers` as an administrative group to be verified during a security audit of the **/usr/bin/abc** file, type:

```
tcbck -a /usr/bin/abc setuids=jfh,jsl setgids=developers
```

After installing a program, you might not know which new files are registered in the **/etc/security/sysck.cfg** file. These files can be found and added with the following command:

```
tcbck -t tree
```

This command string displays the name of any file that is to be registered in the **/etc/security/sysck.cfg** file.

Deleting a trusted program:

If you remove a file from the system that is described in the **/etc/security/sysck.cfg** file, you must also remove the description of this file from the **/etc/security/sysck.cfg** file.

For example, if you have deleted the **/etc/cvid** program, the following command string produces an error message:

```
tcbck -t ALL
```

The resulting error message is as follows:

```
3001-020 The file /etc/cvid was not found.
```

The description for this program remains in the **/etc/security/sysck.cfg** file. To remove the description of this program, type the following command:

```
tcbck -d /etc/cvid
```

Configuring additional trusted options

This section provides information about how to configure additional options for the Trusted Computing Base (TCB).

Restricting access to a terminal:

This section describes how to configure the operating system to restrict terminal access.

The **getty** and **shell** commands change the owner and mode of a terminal to prevent untrusted programs from accessing the terminal. The operating system provides a way to configure exclusive terminal access.

Using the Secure Attention Key:

A trusted communication path is established by pressing the Secure Attention Key (SAK) reserved key sequence (Ctrl-X, and then Ctrl-R).

Note: Use caution when using SAK because it kills all processes that attempt to access the terminal and any links to it (for example, **/dev/console** can be linked to **/dev/tty0**).

A trusted communication path is established under the following conditions:

- When logging in to the system

  After you press the SAK:

  - If a new login screen displays, you have a secure path.
  - If the trusted shell prompt displays, the initial login screen was an unauthorized program that might have been trying to steal your password. Determine who is currently using this terminal by using the **who** command and then log off.
- When you want the command you enter to result in a trusted program running. Some examples of this include:
  - Running as root user. Run as root user only after establishing a trusted communication path. This ensures that no untrusted programs are run with root-user authority.
  - Running the **su**, **passwd**, and **newgrp** commands. Run these commands only after establishing a trusted communication path.

Configuring the Secure Attention Key:

Configure the Secure Attention Key to create a trusted communication path.

Each terminal can be independently configured so that pressing the Secure Attention Key (SAK) at that terminal creates a trusted communication path. This is specified by the **sak_enabled** attribute in **/etc/security/login.cfg** file. If the value of this attribute is True, the SAK is enabled.

If a port is to be used for communications, (for example, by the **uucp** command), the specific port used has the following line in its stanza of the **/etc/security/login.cfg** file:

```
sak_enabled = false
```

This line (or no entry in that stanza) disables the SAK for that terminal.

To enable the SAK on a terminal, add the following line to the stanza for that terminal:

```
sak_enabled = true
```

Controlled Access Protection Profile and Evaluation Assurance Level 4+

Beginning in AIX 5.2, system administrators can install a system with the Controlled Access Protection Profile (CAPP) and Evaluation Assurance Level 4+ (EAL4+) option during a base operating system (BOS) installation. A system with this option has restrictions on the software that is installed during BOS installation, plus network access is restricted.

CAPP/EAL4+ compliant system overview

A CAPP system is a system that has been designed and configured to meet the Controlled Access Protection Profile (CAPP) for security evaluation according to the Common Criteria. The CAPP specifies the functional requirements for the system, similar to the earlier TCSEC C2 standard (also known as the *Orange Book*).

A Common Criteria (CC) Evaluated System is a system that has been evaluated according to the Common Criteria, an ISO standard (ISO 15408) for the assurance evaluation of IT products. The system configuration that meets these requirements is referred to as a *CAPP/EAL4+ system* in this guide.

If a system is evaluated according to the CC, the CC evaluation is valid only for a specific system configuration (hardware and software). Changing the relevant security configuration results in a nonevaluated system. This does not necessarily mean that the security of the system will be reduced, but only indicates that the system is no longer in a certified configuration. Neither the CAPP nor the CC cover all possible security configuration options of AIX 5.2. Some features, such as IPsec or custom-password checking modules, are not included, but can be used to enhance the security of the system.

The AIX 5.2 CAPP/EAL4+ system includes the base operating system on 64-bit POWER3™ and POWER4™ processors with the following:

- Logical Volume Manager (LVM) and the enhanced journaled file system (JFS2)
- The X-Windows system with the CDE interface
- Basic Internet Protocol version 4 (IPv4) network functions (Telnet, FTP, rlogin, rsh/rcp)
- Network File System (NFS)

A CAPP/EAL4+ system is considered to be in a secured state if the following conditions apply:

- If auditing is configured and the system is in multi-user mode, then auditing must be operational.
- The system accepts user logins and services network requests.
- For a distributed system, the administrative databases are NFS-mounted from the master server.

The following administrative interfaces to the security functionality are provided:

- Identification and authentication measures (configuration of users, password settings, login configuration, and so on.)
- Audit measures (configuring bin mode audition, selecting audited events, processing audit trails, and so on.)
- Discretionary access control (permission bits and ACLs for file system objects, IPC mechanisms and TCP ports)
- Setting the system time
- Running the diag diagnostic subsystem

- Running the **su** command to become a privileged administrator (root)

This includes the configuration files and system calls that can be used to perform the appropriate administration.

The following user interfaces to the security functionality are provided:

- The **passwd** command for changing a user's password
- The **su** command for changing a user's identity
- The at, batch, and crontab facilities for the scheduling of command processing
- Discretionary access control (permission bits and ACLs for file system objects and IPC mechanisms)
- Login mechanisms (for example, identification and authentication mechanisms) for the system console and the supported network applications (such as, telnet and ftp)

This includes the system calls dealing with the settings of user identity or access control.

The AIX 5.2 CAPP/EAL4+ system runs on hardware platforms based on IBM® eServer™ pSeries® Symmetric Multiprocessor (SMP) systems using the POWER3-II processor (IBM eServer pSeries 610) with one and two processors, SMP systems using the RS64 IV processor (IBM eServer pSeries 660), and SMP systems using the POWER4 processor (IBM eServer pSeries 690). Peripheral devices that are supported are terminals and printers, hard disks and CD-ROM drives as storage devices, and streamers and diskette drives as backup devices. Supported network connector types are Ethernet and token ring.

Beginning in AIX 5L™ Version 5.2 with the 5200-01 Recommended Maintenance package, the CAPP/EAL4+ technology runs on POWER4 processor (IBM eServer pSeries 630, IBM eServer pSeries 650, and IBM eServer pSeries 690) hardware platforms that support logical partition configuration. Peripheral devices that are supported are terminals and printers, hard disks and CD-ROM drives as storage devices, and streamers and diskette drives as backup devices. Supported network connector types are Ethernet and token ring.

Note: Administrators must inform all users of the system not to use the **$HOME/.rhosts** file for remote login and running commands.

Installing a CAPP/EAL4+ system

To set the CAPP/EAL4+ option during a BOS installation, do the following:

1. In the Installation and Settings screen, select **More Options**.
2. In the More Options screen, type the number corresponding to the Yes or No choice for **Enable CAPP and EAL4+ Technology**. The default is set to `No`.

The **Enable CAPP and EAL4+ Technology** option is available only under the following conditions:

- The installation method is set to new and complete overwrite installation.
- The English language is selected.
- The 64-bit kernel is enabled.
- The enhanced journaled file system (JFS2) is enabled.

When the **Enable CAPP and EAL4+ Technology** option is set to Yes, the **Trusted Computing Base** option is also set to Yes, and the only valid **Desktop** choices are NONE or CDE.

If you are performing a nonprompted installation using a customized **bosinst.data** file, the INSTALL_TYPE field must be set to `CC_EVAL` and the following fields must be set as follows:

```
control_flow:
 CONSOLE = ???
 PROMPT = yes
 INSTALL_TYPE = CC_EVAL
 INSTALL_METHOD = overwrite
 TCB = yes
 DESKTOP = NONE or CDE
 ENABLE_64BIT_KERNEL = yes
 CREATE_JFS2_FS = yes
 ALL_DEVICES_KERNELS = no
 MOZILLA_BUNDLE = no
 HTTP_SERVER_BUNDLE = no
 KERBEROS_5_BUNDLE = no
 SERVER_BUNDLE = no
 ALT_DISK_INSTALL_BUNDLE = no

locale:
 CULTURAL_CONVENTION = en_US or C
 MESSAGES = en_US or C
```

CAPP/EAL4+ and the Network Installation Management environment

Installation of CAPP/EAL4+ technology clients can be performed using the Network Installation Management (NIM) environment.

The NIM master is configured to provide the resources needed to install the appropriate CAPP/EAL4+ level of AIX 5L. NIM clients may then be installed using the resources located on the NIM master. You can perform a nonprompted NIM installation of the client by setting the following fields in the bosinst_data resource:

```
control_flow:
 CONSOLE = ???
 PROMPT = no
 INSTALL_TYPE = CC_EVAL
 INSTALL_METHOD = overwrite
 TCB = yes
 DESKTOP = NONE or CDE
 ENABLE_64BIT_KERNEL = yes
 CREATE_JFS2_FS = yes
 ALL_DEVICES_KERNELS = no
 MOZILLA_BUNDLE = no
 HTTP_SERVER_BUNDLE = no
 KERBEROS_5_BUNDLE = no
 SERVER_BUNDLE = no
 ALT_DISK_INSTALL_BUNDLE = no

locale:
 CULTURAL_CONVENTION = en_US or C
 MESSAGES = en_US or C
```

The NIM master cannot be configured as a CAPP/EAL4+ system and cannot be connected to the same network with other CAPP/EAL4+ systems. When initiating the installation from the NIM master, the **Remain NIM client after install SMIT** menu option must be set to No. After a NIM client is installed as a CAPP/EAL4+ system, the NIM client must be removed from the NIM master's network, and additional software installations and updates cannot be performed using the NIM master.

An example situation is to have two network environments; the first network consists of the NIM master and the non-CAPP/EAL4+ systems; the second network consists only of CAPP/EAL4+ systems. Perform the NIM installation on the NIM client. After the installation has completed, disconnect the newly installed CAPP/EAL4+ system from the NIM master's network and connect the system to the evaluated network.

A second example consists of one network. The NIM master is not connected to the network when other systems are operating in the evaluated configuration, and CAPP/EAL4+ systems are not connected to the network during NIM installation.

CAPP/EAL4+ software bundle

When the **CAPP/EAL4+** option is selected, the contents of the **/usr/sys/inst.data/sys_bundles/CC_EVAL.BOS.autoi** installation bundle are installed.

You can optionally select to install the graphics software bundle and the documentation services software bundle with the **CAPP/EAL4+** option selected. If you select the **Graphics Software** option with the **CAPP/EAL4+** option, the contents of the **/usr/sys/inst.data/sys_bundles/CC_EVAL.Graphics.bnd** software bundle are installed. If you select the Documentation Services Software option with the **CAPP/EAL4+** option, the contents of the **/usr/sys/inst.data/sys_bundles/CC_EVAL.DocServices.bnd** software bundle are installed.

After the Licensed Program Products (LPPs) have been installed, the system changes the default configuration to comply with the CAPP/EAL4+ requirements. The following changes are made to the default configuration:

- Remove `/dev/echo` from the **/etc/pse.conf** file.
- Instantiate streams devices.
- Allow only root to access removable media.
- Remove non-CC entries from the **inetd.conf** file.
- Change various file permissions.
- Register symbolic links in the **sysck.cfg** file.
- Register devices in the **sysck.cfg** file.
- Set default user and port attributes.
- Configure the doc_search application for browser use.
- Remove httpdlite from the **inittab** file.
- Remove writesrv from the **inittab** file.
- Remove mkatmpvc from the **inittab** file.
- Remove atmsvcd from the **inittab** file.
- Disable snmpd in the **/etc/rc.tcpip** file.
- Disable hostmibd in the **/etc/rc.tcpip** file.
- Disable snmpmibd in the **/etc/rc.tcpip** file.
- Disable aixmibd in the **/etc/rc.tcpip** file.
- Disable muxatmd in the **/etc/rc.tcpip** file.
- NFS port (2049) is a privileged port.
- Add missing events to the **/etc/security/audit/events** file.
- Ensure that the loopback interface is running.
- Create synonyms for **/dev/console**.
- Enforce default X-server connection permissions.
- Change the **/var/docsearch** directory so that all files are world-readable.
- Add Object Data Manager (ODM) stanzas to set the console permissions.
- Set permissions on BSD-style ptys to 000.
- Disable **.netrc** files.
- Add patch directory processing.

Graphical user interface

The CAPP/EAL4+ compliant system includes the X Windows® System as a graphical user interface.

X Windows provides a mechanism for displaying graphical clients, such as clocks, calculators, and other graphical applications, as well as multiple terminal sessions using the **aixterm** command. The X Windows System is started with the **xinit** command from the initial command line after a user has logged in at the host's console.

To start an X Windows session, type:

```
xinit
```

This command starts the X Windows server with local access mechanisms enabled for the invoker only. X Windows clients that are set-UID to root will be able to access the X Windows server via the UNIX® domain socket using the root override on the access restrictions. X Windows clients that are set-UID to other users or that are started by other users will not be able to access the X Windows server. This restriction prevents other users of a host from gaining unauthorized access to the X Windows server.

CAPP/EAL4+ system physical environment

The CAPP/EAL4+ system has specific requirements for the environment in which it is run.

The requirements are as follows:

- Physical access to the systems must be restricted so that only authorized administrators can use the system consoles.
- The Service Processor is not connected to a modem.
- Physical access to the terminals is restricted to authorized users.
- The physical network is secure against eavesdropping and spoofing programs (also called Trojan horse programs). When communicating over insecure lines, additional security measures, such as encryption, are needed.
- Communication with other systems that are not AIX 5.2 CAPP/EAL4+ systems, or are not under the same management control, is not permitted.
- Only IPv4 is to be used when communicating with other CAPP/EAL4+ systems, IPv6 has not been evaluated.
- Users must not be allowed to change the system time.
- Systems in an LPAR environment cannot share PHBs.

CAPP/EAL4+ system organizational environment

Certain procedural and organizational requirements must be met for a CAPP/EAL4+ system.

The following requirements must be met:

- Administrators must be trustworthy and well trained.
- Only users authorized to work with the information on the systems are granted user IDs on the system.
- Users must use high-quality passwords (as random as possible and not affiliated with the user or the organization). For information about setting up password rules, see “Passwords” on page 38.
- Users must not disclose their passwords to others.
- Administrators must have sufficient knowledge to manage security critical systems.
- Administrators must work in accordance with the guidance provided by the system documentation.
- Administrators must log in with their personal ID and use the **su** command to switch to superuser mode for administration.
- Passwords generated for system users by administrators must be transmitted securely to the users.
- Those who are responsible for the system must establish and implement the necessary procedures for the secure operation of the systems.
- Administrators must ensure that the access to security-critical system resources is protected by appropriate settings of permission bits and ACLs.
- The physical network must be approved by the organization to carry the most sensitive data held by the systems.
- Maintenance procedures must include regular diagnostics of the systems.
- Administrators must have procedures in place that ensure a secure operation and recovery after a system failure.
- The *LIBPATH* environment variable should not be changed, because this might result in a trusted process loading an untrusted library.
- Wiretapping and trace software (tcpdump, trace) must not be used on an operational system.

- Anonymous protocols such as HTTP may only be used for public information (for example, the online documentation).
- Only TCP-based NFS can be used.
- Access to removable media is not to be given to users. The device files are to be protected by appropriate permission bits or ACLs.
- Only root authority is used when administering AIX. None of the role-based and group-based administration-delegation features, nor the privilege mechanism of AIX, are included in the CAPP/EAL4+ compliance.
- Administrators must not use dynamic partitioning to allocate and deallocate resources. Partition configuration may only be performed while no partitions at all are running.

CAPP/EAL4+ system operational environment

Certain operational requirements and procedures must be met for a CAPP/EAL4+ system.

The following requirements and procedures must be met:

- If using a Hardware Management Console (HMC), the HMC is located in a physically controlled environment.
- Only authorized personnel can access to the operational environment and the HMC.
- If using an HMC, the HMC can only be used for the following tasks:
  - Initial configuration of the partitions. A partition cannot be active during the configuration process.
  - Restarting of ″hanging″ partitions
- The HMC must not be used throughout operation of the configured system.
- The system's ″call home″ feature must be disabled.
- Remote modem access to the system must be disabled.
- If AIX runs in an LPAR-enabled environment, the administrator should check with the LPAR documentation for requirements on the EAL4+ operation of logical partitions.
- The service authority feature must be disabled on logical partitions.

CAPP/EAL4+ system configuration

This section provides Information about the configuration of the Controlled Access Protection Profile (CAPP) and Evaluation Assurance Level 4+ (EAL4+) system.

Administration:

Administrators must log in with their personal user account and use the **su** command to become the root user for the administration of the system.

To effectively prevent guessing the root account's password, allow only authorized administrators to use the **su** command on the root account. To ensure this, do the following:

1. Add an entry to the **root** stanza of the **/etc/security/user** file as follows:

   ```
   root:
     admin = true
     .
     .
     .
     sugroups = SUADMIN
   ```

2. Define group in the **/etc/group** file containing only the user IDs of authorized administrators as follows:

   ```
   system:!:0:root,paul
   staff:!:1:invscout,julie
   bin:!:2:root,bin
   .
   .
   .
   SUADMIN:!:13:paul
   ```

Administrators must also adhere to the following procedures:

- Establish and implement procedures to ensure that the hardware, software and firmware components that comprise the distributed system are distributed, installed, and configured in a secure manner.
- Ensure that the system is configured so that only an administrator can introduce new trusted software into the system.
- Implement procedures to ensure that users clear the screen before logging off from serial login devices (for example, IBM 3151 terminals).

User and port configuration:

AIX configuration options for users and ports must be set to satisfy the requirements of the evaluation. The actual requirement is that the probability of correctly guessing a password should be at least 1 in 1,000,000, and the probability of correctly guessing a password with repeated attempts in one minute should be at least 1 in 100,000.

The **/etc/security/user** file shown in the following example uses the **/usr/share/dict/words** dictionary list. The **/usr/share/dict/words** file is contained in the **bos.data** fileset. You must install the **bos.data** fileset prior to configuring the **/etc/security/user** file. The recommended values for the **/etc/security/user** file are the following:

```
default:
  admin = false
  login = true
  su = true
  daemon = true
  rlogin = true
  sugroups = ALL
  admgroups =
  ttys = ALL
  auth1 = SYSTEM
  auth2 = NONE
  tpath = nosak
  umask = 077
  expires = 0
  SYSTEM = "compat"
  logintimes =
  pwdwarntime = 5
  account_locked = false
  loginretries = 3
  histexpire = 52
  histsize = 20
  minage = 0
  maxage = 8
  maxexpired = 1
  minalpha = 2
  minother = 2
  minlen = 8
  mindiff = 4
  maxrepeats = 2
  dictionlist = /usr/share/dict/words
  pwdchecks =
  dce_export = false

root:
  rlogin = false
  login = false
```

The default settings in the **/etc/security/user** file should not be overwritten by specific settings for single users.

Note: Setting `login = false` in the root stanza prevents direct root login. Only user accounts that have **su** privileges for the root account will be able to log in as the root account. If a Denial of Service attack is launched against the system that sends incorrect passwords to the user accounts, it could lock all the user accounts. This attack might prevent any user (including administrative users) from logging into the system. Once a 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 the `loginretries` user attribute. If all the administrative accounts become locked, you might need to reboot the system into maintenance mode and run the **chsec** command. For more information about using the **chsec** command, see "User account control" on page 31.

The suggested values for the **/etc/security/login.cfg** file are the following:

```
default:
  sak_enabled = false
  logintimes =
  logindisable = 4
  logininterval = 60
  loginreenable = 30
  logindelay = 5
```

Resource limits:

When setting resource limits in the **/etc/security/limits** file, make sure that the limits correspond to the needs of the processes on the system.

In particular, the `stack` and `rss` sizes should *never* be set to `unlimited`. An unlimited stack might overwrite other segments of the running process, and an unlimited `rss` size allows a process to use all real memory, therefore creating resource problems for other processes. The `stack_hard` and `rss_hard` sizes should also be limited.

Audit subsystem:

The following procedures help protect the audit subsystem:

- Configure the audit subsystem to record all the relevant security activities of the users. To ensure that the file space needed for auditing is available and is not impaired by other consumers of file system space, set up a dedicated file system for audit data.
- Protect audit records (such as audit trails, bin files, and all other data stored in **/audit**) from non-root users.
- For the CAPP/EAL4+ system, **bin** mode auditing must be set up when the audit subsystem is used. For information about how to set up the audit subsystem, refer to "Setting up auditing" on page 65.
- At least 20 percent of the available disk space in a system should be dedicated to the audit trail.
- If auditing is enabled, the *binmode* parameter in the start stanza in the **/etc/security/audit/config** file should be set to `panic`. The *freespace* parameter in the bin stanza should be configured at minimum to a value that equals 25 percent of the disk space dedicated to the storage of the audit trails. The *bytethreshold* and *binsize* parameters should each be set to 65 536 bytes.
- Copy audit records from the system to permanent storage for archival.

Network configuration:

Network configuration must use Discretionary Access Control for Internet Ports (DACinet) to make sure that the X protocol (X11) and NFS cannot be used anonymously.

The **dacinet** command prevents the following conditions:

- A user from taking over another user's desktop with X11.
- A user on a client from forging requests to an NFS server that would permit the user to become root. Normally, a user accesses a remote NFS server by making requests to the Logical File System on the

local host, which then makes the request (as root) to the remote server. Setting an ACL for root only and not permitting this port to be bypassed ensures that the user cannot send direct protocol requests to an NFS server.

For more information about the **dacinet** command, see “User based TCP port access control with discretionary access control for internet ports” on page 134.

System services:

The following table is a list of standard system services on a Controlled Access Protection Profile (CAPP) and Evaluation Assurance Level 4+ (EAL4+) system.

This table shows the standard system services running on a CAPP/EAL4+ system (if there is no graphics card).

Table 1. Standard System Services

UID	Command	Description
root	/etc/init	Init process
root	/usr/sbin/syncd 60	File system sync daemon
root	/usr/sbin/srcmstr	SRC master daemon
root	/usr/sbin/cron	CRON facility with AT® support
root	/usr/ccs/bin/shlap64	Shared Library Support Daemon
root	/usr/sbin/syslogd	Syslog daemon
root	/usr/lib/errdemon	AIX error log daemon
root	/usr/sbin/getty /dev/console	getty / TSM
root	/usr/sbin/portmap	Portmapper for NFS and CDE
root	/usr/sbin/biod 6	NFS Client
root	/usr/sbin/rpc.lockd	NFS lock daemon
daemon	/usr/sbin/rpc.statd	NFS stat daemon
root	/usr/sbin/rpc.mountd	NFS mount daemon
root	/usr/sbin/nfsd	NFS server daemon
root	/usr/sbin/inetd	Inetd master daemon
root	/usr/sbin/uprintfd	Kernel print daemon
root	/usr/sbin/qdaemon	Queuing daemon
root	/usr/lpp/diagnostics/bin/diagd	Diagnostics

Running a CAPP/EAL4+ distributed system:

To run a distributed system that is CAPP/EAL4+ compliant, all users must have identical user IDs on all systems. Although this can be achieved with NIS, the result is not secure enough for a CAPP/EAL4+ system.

This section describes a distributed setup that ensures that the user IDs are identical on all systems that are CAPP/EAL4+ compliant.

The master system stores the identification and authentication data (user and group configuration) for the whole distributed system. All other systems use NFS to mount this data. NFS is protected by DACinet so that only the administrators can access the NFS ports on the master.

Authentication data can be changed by any administrator by using tools, such as SMIT, on any system. Authentication data is physically changed on the master.

All shared identification and authentication data comes from the **/etc/data.shared** directory. The regular identification and authentication files are replaced by symbolic links into the **/etc/data.shared** directory.

Shared files in the distributed system:

The following files are shared in the distributed system. Typically, they come from the **/etc/security** directory.

/etc/group
The **/etc/group** file

/etc/hosts
The **/etc/hosts** file

/etc/passwd
The **/etc/passwd** file

/etc/security/.ids
The next available user and group ID

/etc/security/.profile
The default **.profile** file for new users

/etc/security/acl
The **/etc/security/acl** file stores system-wide ACL definitions for protected services that will be reactivated at the next system boot by the **/etc/rc.tcpip** file.

/etc/security/audit/bincmds
Bin-mode auditing commands for this host

/etc/security/audit/config
Local audit configuration

/etc/security/audit/events
List of audit events and formats

/etc/security/audit/objects
List of audited objects on this host

/etc/security/audit/streamcmds
Stream-mode auditing commands for this host

/etc/security/environ
Per-user environmental variables

/etc/security/group
Extended group information from the **/etc/security/group** file

/etc/security/limits
Per-user resource limits

/etc/security/passwd
Per-user passwords

/etc/security/priv
Ports that are to be designated as privileged when the system starts are listed in the **/etc/security/priv** file

/etc/security/services
Ports listed in the **/etc/security/services** file are considered exempt from ACL checks

/etc/security/user
Per-user and default user attributes

Nonshared files in the distributed system:

The following files in the **/etc/security** directory are not to be shared in the distributed system, but are to remain host-specific:

/etc/security/failedlogin
Log file for failed logins per host

/etc/security/lastlog
Per-user information about the last successful and unsuccessful logins on this host

/etc/security/login.cfg
Host-specific login characteristics for trusted path, login shells, and other login-related information

/etc/security/portlog
Per-port information for locked ports on this host

The automatically generated backup files of the shared files are also nonshared. Backup files have the same name as the original file, but have a lowercase letter `o` prepended.

Setting up the distributed system (Master):

On the master, a new logical volume is created that holds the file system for the identification and authentication data. The logical volume is named **/dev/hd10sec** and it is mounted on the master system as **/etc/data.master**.

To generate the necessary changes on the master system, run the **mkCCadmin** command with the IP address and host name of the master, as follows:

```
mkCCadmin -m -a ipaddress hostname
```

Setting up the distributed system (all systems):

Setting up the distributed system for all systems.

All data that is to be shared is moved to the **/etc/data.shared** directory. At startup, all systems will mount the master's **/etc/data.master** directory over the **/etc/data.shared** directory. The master itself uses a loopback mount.

Client systems are set up by running the following:

```
mkCCadmin -a ipaddress hostname
```

To change the client to use a different master, use the **chCCadmin** command.

After a system has been integrated into the distributed identification and authentication system, the following additional inittab entries are generated:

isCChost
Initializes the system to CAPP/EAL4+ mode.

rcCC Clears all DACinet ACLs and opens only the ports needed for the portmapper and NFS. It then mounts the shared directory.

rcdacinet
Loads additional DACinet ACLs that the administrator might have defined.

When running the distributed system, consider the following:

- Administrators must make sure that the shared data is mounted before changing shared configuration files to ensure that the shared data is seen on all systems.
- Changing the root password is the only administrative action that is permitted while the shared directory is not mounted.

Using the DACinet feature for user-based and port-based network access control:

The DACinet feature can be used to restrict the access of users to TCP ports.

For more information about DACinet, see “User based TCP port access control with discretionary access control for internet ports” on page 134. For example, when using DACinet to restrict access to port TCP/25 inbound to root only with the DACinet feature, only root users from CAPP/EAL4+ compliant hosts can access this port. This situation limits the possibility of regular users spoofing e-mail by using telnet to connect to port TCP/25 on the victim.

To activate the ACLs for TCP connections at boot time, the **/etc/rc.dacinet** script is run from **/etc/inittab**. It will read the definitions in the **/etc/security/acl** file and load ACLs into the kernel. Ports which should not be protected by ACLs should be listed in the **/etc/security/services** file, which uses the same format as the **/etc/services** file.

Assuming a subnet of 10.1.1.0/24 for all the connected systems, the ACL entries to restrict access to the root user only for X (TCP/6000) in the **/etc/security/acl** file would be as follows:

```
6000    10.1.1.0/24 u:root
```

Installing additional software on a CAPP/EAL4+ compliant system:

The administrator can install additional software on the CAPP/EAL4+ compliant system. If the software is not run by the root user or with root-user privileges, this will not invalidate the CAPP/EAL4+ compliance. Typical examples include office applications that are run only by regular users and have no SUID components.

Additionally, installed software that runs with root-user privileges invalidates the CAPP/EAL4+ compliance. This means, for example, drivers for the older JFS should not be installed, as they are running in kernel mode. Additional daemons that are run as root (for example, an SNMP daemon) also invalidates the CAPP/EAL4+ compliance. A CAPP/EAL4+ enabled system cannot be upgraded (normally).

A CAPP/EAL4+ compliant system is rarely used in the evaluated configuration, especially in a commercial environment. Typically, additional services are needed, so that the production system is based on an evaluated system, but does not comply with the exact specification of the evaluated system.

Login control

Changing the login screen defaults after a system installation for security reasons.

Potential hackers can get valuable information from the default AIX login screen, such as the host name and the version of the operating system. This information would allow them to determine which exploitation methods to attempt. For security reasons, you may want to change the login screen defaults as soon as possible after a system installation.

The KDE and GNOME desktops share some of the same security issues. For more information about KDE and GNOME, refer to the *AIX 5L Version 5.3 Installation Guide and Reference*.

For information about users, groups, and passwords, see “Users, roles, and passwords” on page 24.

Setting up login controls

Setting up login controls in the **/etc/security/login.cfg** file.

To make it harder to attack a system with password guessing, set up login controls in the **/etc/security/login.cfg** file as follows:

Table 2. Attributes and Recommended Values for Login Control.

Attribute	Applies to PtYs (Network)	Applies to TTYs	Recommended Value	Comments
sak_enabled	Y	Y	false	The Secure Attention key is rarely needed. See “Using the Secure Attention Key” on page 5.
logintimes	N	Y		Specify allowed login times here.
logindisable	N	Y	4	Disable login on this terminal after 4 consecutive failed attempts.
logininterval	N	Y	60	Terminal will be disabled when the specified invalid attempts have been made within 60 seconds.
loginreenable	N	Y	30	Re-enable the terminal after it was automatically disabled after 30 minutes.
logindelay	Y	Y	5	The time in seconds between login prompts. This will be multiplied with the number of failed attempts; for example, 5,10,15,20 seconds when 5 is the initial value.

These port restrictions work mostly on attached serial terminals, not on pseudo-terminals used by network logins. You can specify explicit terminals in this file, for example:

```
/dev/tty0:
        logintimes = 0600-2200
        logindisable = 5
        logininterval = 80
        loginreenable = 20
```

Changing the welcome message on the login screen

To prevent displaying certain information on login screens, edit the *herald* parameter in the /etc/security/login.cfg file.

The default *herald* contains the welcome message that displays with your login prompt. To change this parameter, you can either use the **chsec** command or edit the file directly.

The following example uses the **chsec** command to change the default *herald* parameter:

```
# chsec -f /etc/security/login.cfg -a default -herald
"Unauthorized use of this system is prohibited.\n\nlogin: "
```

For more information about the **chsec** command, see the *AIX 5L Version 5.3 Commands Reference, Volume 1*.

To edit the file directly, open the **/etc/security/login.cfg** file and update the *herald* parameter as follows:

```
default:
herald ="Unauthorized use of this system is prohibited\n\nlogin:"
 sak_enable = false
 logintimes =
 logindisable = 0
 logininterval = 0
 loginreenable = 0
 logindelay = 0
```

Note: To make the system more secure, set the *logindisable* and *logindelay* variables to a number greater than `0` (`# > 0`).

Changing the login screen for the common desktop environment

This security issue also affects the Common Desktop Environment (CDE) users. The CDE login screen also displays, by default, the host name and the operating system version. To prevent this information from being displayed, edit the **/usr/dt/config/$LANG/Xresources** file, where **$LANG** refers to the local language installed on your machine.

In our example, assuming that **$LANG** is set to **C**, copy this file into the **/etc/dt/config/C/Xresources** directory. Next, open the **/usr/dt/config/C/Xresources** file and edit it to remove welcome messages that include the host name and operating system version.

For more information about CDE security issues, see “Managing X11 and CDE concerns” on page 23.

Disabling the display of the user name and changing the password prompt

In a secure environment, it might be necessary to hide the display of the login user name or to provide a custom password prompt that differs from the default.

The default message behavior for the login and password prompt is shown below:

```
login: foo
foo's Password:
```

To disable the display of the user name from prompts and system error messages, edit the *usernameecho* parameter in the **/etc/security/login.cfg** file. The default value for *usernameecho* is true which results in the user name being displayed. To change this parameter, you can either use the **chsec** command or edit the file directly.

The following example uses the **chsec** command to change the default *usernameecho* parameter to false:

```
# chsec -f /etc/security/login.cfg -s default -a usernameecho=false
```

For more information about the **chsec** command, see the *AIX 5L Version 5.3 Commands Reference, Volume 1*.

To edit the file directly, open the **/etc/security/login.cfg** file and add or modify the *usernameecho* parameter as follows:

```
default:
 usernamecho = false
```

Setting the *usernameecho* parameter to false will result in the user name not being displayed at the login prompt. Instead, the user name is masked out with ’*’ characters for system prompts and error messages as show below:

```
login:
***'s Password:
```

The password prompt may be separately modified to be a custom string by setting the *pwdprompt* parameter in the **/etc/security/login.cfg** file. The default value is a string ″*user*’s Password: ″ where *user* is replaced with the authenticating user name.

To change this parameter, you can either use the **chsec** command or edit the file directly.

The following example uses the **chsec** command to change the default *pwdprompt* parameter to ″Password: ″:

```
# chsec -f /etc/security/login.cfg -s default -a pwdprompt="Password: "
```

To edit the file directly, open the **/etc/security/login.cfg** file and add or modify the *pwdprompt* parameter as follows:

```
default:
 pwdprompt = "Password: "
```

Setting the *pwdprompt* parameter to ″Password: ″ will result in the specified prompt being displayed by login as well as by other applications that use the system password prompt. The prompt behavior for the login when the a custom prompt has been configured is as follows:

```
login: foo
Password:
```

Setting up system default login parameters

Edit the **/etc/security/login.cfg** file to set up system default login parameters.

To set up base defaults for many login parameters, such as those you might set up for a new user (number of login retries, login re-enable, and login internal), edit the **/etc/security/login.cfg** file.

Securing unattended terminals

Use the **lock** and **xlock** commands to secure your terminal.

All systems are vulnerable if terminals are left logged in and unattended. The most serious problem occurs when a system manager leaves a terminal unattended that has been enabled with root authority. In general, users should log out any time they leave their terminals. Leaving system terminals unsecure poses a potential security hazard. To lock your terminal, use the **lock** command. If your interface is AIXwindows, use the **xlock** command.

Enforcing automatic logoff

Enable automatic logoff to prevent an intruder from compromising the security of the system.

Another valid security concern results from users leaving their accounts unattended for a lengthy period of time. This situation allows an intruder to take control of the user's terminal, potentially compromising the security of the system.

To prevent this type of potential security hazard, you can enable automatic logoff on the system. To do this, edit the **/etc/security/.profile** file to include an automatic logoff value for *all* users, as in the following example:

```
TMOUT=600 ; TIMEOUT=600 ; export readonly TMOUT TIMEOUT
```

The number 600, in this example, is in seconds, which is equal to 10 minutes. However, this method will only work from the shell.

While the previous action allows you to enforce an automatic logoff policy for all users, system users can bypass some restrictions by editing their individual **.profile** files. To completely implement an automatic logoff policy, take authoritative action by providing users with appropriate **.profile** files, preventing write-access rights to these files.

Stack Execution Disable protection

Keeping computer systems secure forms an important aspect of an On Demand business. In today's world of highly networked environments, it has become an extreme challenge to ward off attacks from a variety of sources.

There is increasing likelihood of computer systems falling prey to sophisticated attacks, resulting in disruption to the daily operations of businesses and government agencies. While no security measure can provide foolproof protection against attacks, you should deploy multiple security mechanisms to thwart security attacks. This section covers a security mechanism that is used with AIX to thwart attacks due to buffer overflow based execution.

Security breaches occur in many forms, but one of the most common methods is to monitor the system-provided administrative tools, look for, and exploit buffer overflows. Buffer overflow attacks occur when an internal program buffer is overwritten because data was not properly validated (such as command line, environmental variable, disk or terminal I/O). Attack code is inserted into a running process through the buffer overflow, changing the execution path of the running process. The return address is overwritten and redirected to the inserted-code location. Common causes of breaches include improper or nonexistent bounds checking, or incorrect assumptions about the validity of data sources. For example, a buffer overflow can occur when a data object is large enough to hold 1 KB of data, but the program does not check the bounds of the input and hence can be made to copy more than 1 KB into that data object.

The intruder's goal is to attack a command and/or tool that provides root privileges to a regular user. Control of the program is gained with all the privileges enabled, allowing overflow of the buffers. Attacks are typically focused on a root owned UID set or programs leading to the execution of a shell, thereby gaining root-based shell access to the system.

You can prevent these attacks by blocking execution of attack code entering through the buffer overflow. Disable execution on the memory areas of a process where execution commonly does not take place (stack and heap memory areas).

SED buffer overflow protection mechanism

AIX has enabled the stack execution disable (SED) mechanism to disable the execution of code on a stack and select data areas of a process.

By disabling the execution and then terminating, an infringing program, the attacker is prevented from gaining root user privileges through a buffer overflow attack. While this feature does not stop buffer overflows, it provides protection by disabling the execution of attacks on buffers that have been overflowed.

Beginning with the POWER4 family of processors, you can use a page-level execution enable and/or disable feature for the memory. The AIX SED mechanism uses this underlying hardware support for implementing a no-execution feature on select memory areas. Once this feature is enabled, the operating system checks and flags various files during the executable programs. It then alerts the operating system memory manager and the process managers that the SED is enabled for the process being created. The select memory areas are marked for no-execution. If any execution occurs on these marked areas, the hardware raises an exception flag and the operating system stops the corresponding process. The exception and application termination details are captured through the AIX error log events.

SED is implemented mainly through the **sedmgr** command. The **sedmgr** command allows control of the systemwide SED mode of operation as well as setting the executable file based SED flags.

SED modes and monitoring

The stack execution disable (SED) mechanism in AIX is implemented through systemwide mode flags, as well as individual executable file-based header flags.

While systemwide flags control the systemwide operation of the SED, file level flags indicate how files should be treated in SED. The buffer overflow protection (BOP) mechanism provides for four systemwide modes of operation:

off The SED mechanism is turned off and no process is marked for SED protection.

select Only a select set of files are enabled and monitored for SED protection. The select set of files are chosen by reviewing the SED related flags in the executable program binary headers. The executable program header enables SED related flags to request to be included in the **select** mode.

setidfiles
Allows you to enable SED, not only for the files requesting such a mechanism, but all the important **setuid** and **setgid** system files. In this mode, the operating system not only provides

SED for the files with the **request** SED flag set, but also enables SED for the executable files with the following characteristics (except the files marked for *exempt* in their file headers):

- SETUID files owned by root
- SETGID files with primary group as **system** or **security**

all All executable programs loaded on the system are SED protected except for the files requesting an exemption from SED mode. Exemption related flags are part of the executable program headers.

The SED feature on AIX also provides the ability to monitor instead of stopping the process when an exception happens. This systemwide control allows a system administrator to check for breakdowns and issues in the system environment by monitoring it before the SED is deployed in the production systems.

The **sedmgr** command provides an option that allows you to enable SED to monitor files instead of stopping the processes when exceptions occur. The system administrator can evaluate whether an executable program is doing any legitimate stack execution. This setting works in conjunction with the systemwide mode set using the `-c` option. When the **monitor** mode is turned on, the system allows the process to continue operating even if an SED-related exception occurs. Instead of stopping the process, the operating system logs the exception in the AIX error log. If SED monitoring is off, the operating system stops any process that violates and raises an exception per SED facility.

Any changes to the SED mode systemwide flags requires that you restart the system for the changes to take effect. All of these types of events are audited.

SED flags for executables

In AIX, you can use the **sedmgr** command to flag executables from the SE mechanism.

Linker has been enhanced to support two new SED related flags to enable `select` and `exempt` options in the executable's headers. The **select** flag allows for an executable to request and be part of SED protection during the `select` mode of systemwide SED operation, whereas the **exempt** flag allows an executable to request for an exemption from the SED mechanism. These executables are not enabled for execution disable on any of the process memory areas.

The exemption flag allows a system administrator to monitor the SED mechanism, and evaluate the situation. The system administrator can enable execution on stack and data areas as necessary for the application, with the associated risks understood.

The following table shows how the systemwide settings and file settings affect the SED mode of operation:

Table 3. Systemwide settings and file settings affecting the SED mode

	Executable file SED flags			
System SED mode	**request**	**exempt**	**system**	**Setuid-root or setgid-system/security files**
off	–	–	–	–
select	enabled	–	–	–
setgidfiles	enabled	–	–	enabled
all	enabled	–	enabled	enabled

SED issues and considerations

By default, AIX SED is shipped in **select** mode. A number of **setuid** and **setgid** programs are **select**-enabled for SED and operate in protected mode by default.

SED enablement might cause older binary files to break if they are not capable of handling the no-execution feature on the stack heap areas. These applications must run on stack data areas. The system administrator can evaluate the situation and flag the file for an exemption using the **bopmgr** command. AIX Java™ 1.3.1 and AIX Java 1.4.2 have Just-In-Time (JIT) compilers that dynamically generate and execute native object code while running Java applications (the Java Virtual Machine decides which code to compile based on the execution profile of the application). This object code is stored in data buffers allocated by the JIT. Consequently, if AIX is configured to run in the SED **ALL** mode, the system administrator must set the Java binary file's exemption flag.

When SED-related flags in an executable file are changed, they apply only to a future load and execution of the file. This change does not apply to currently operating processes based on this file. The SED facility controls and monitors both 32- and 64-bit executable programs for the systemwide and file-level settings. The SED facility is available only when the AIX operating system is used with the 64-bit kernel.

Related information

sedmgr command

AIX **Error-Logging Facility**

Managing X11 and CDE concerns

This section discusses potential security vulnerabilities involved with the X11 X server and the Common Desktop Environment (CDE).

Removing the /etc/rc.dt file

Remove the **/etc/rc.dt** file on systems that require a high level of security.

Although running the CDE interface is convenient for users, security issues are associated with it. For this reason, do not run CDE on servers that require a high level of security. The best solution is to avoid installing CDE (dt) file sets. If you have installed these file sets on your system, consider uninstalling them, especially the **/etc/rc.dt** script, which starts CDE.

For more information about CDE, see the *AIX 5L Version 5.3 System Management Guide: Operating System and Devices*.

Preventing unauthorized monitoring of remote X server

An important security issue associated with the X11 server is unauthorized silent monitoring of a remote server.

The **xwd** and **xwud** commands can be used to monitor X server activity because they have the ability to capture keystrokes, which can expose passwords and other sensitive data. To solve this problem, remove these executable files unless they are necessary under your configuration, or, as an alternative, change access to these commands to be root only.

The **xwd** and **xwud** commands are located in the **X11.apps.clients** fileset.

If you do need to retain the **xwd** and **xwud** commands, consider using OpenSSH or MIT Magic Cookies. These third-party applications help prevent the risks that are created by running the **xwd** and **xwud** commands.

For more information about OpenSSH and MIT Magic Cookies, refer to each application's respective documentation.

Enabling and disabling access control

The X server allows remote hosts to use the **xhost +** command to connect to your system.

Ensure that you specify a host name with the **xhost +** command, because it disables access control for the X server. This allows you to grant access to specific hosts, which eases monitoring for potential attacks to the X server. To grant access to a specific host, run the **xhost** command as follows:

```
# xhost + hostname
```

If you do not specify a host name, access will be granted to all hosts.

For more information about the **xhost** command, see the *AIX 5L Version 5.3 Commands Reference*

Disabling user permissions to run the xhost command

You can prevent the unauthorized execution of the **xhost** command by using the **chmod** command.

Another way to ensure that the **xhost** command is being used appropriately is to restrict execution of this command to root-user authority only. To do this, use the **chmod** command to change the permissions of **/usr/bin/X11/xhost** to 744, as follows:

```
chmod 744/usr/bin/X11/xhost
```

Users, roles, and passwords

This section describes how to manage AIX users and roles.

Account ID

Each user account has a numeric ID which uniquely identifies the account. AIX grants authorization according to Account ID.

It is important to understand that accounts with the same ID are virtually the same account. When creating users and groups, the AIX **mkuser** and **mkgroup** commands always check for the target registry to make sure that the account to be created has no ID collision with existing accounts.

The system can also be configured to check all user (group) registries during account creation using the **dist_uniqid** system attribute. The **dist_uniqid** attribute of the usw stanza in the **/etc/security/login.cfg** file can be managed using the **chsec** command. To configure the system to always check for id collision against all registries, run:

```
# chsec -f /etc/security/login.cfg -s usw -a dist_uniqid=always
```

The **dist_uniqid** attribute allows three valid values:

never This value does not check for ID collision against the non-target registries (default).

always
This value checks for ID collision against all other registries. If collision detected between the target registry and any other registry, the **mkuser** (**mkgroup**) command picks a unique ID which is not used by any registry. It only fails if the ID value is specified from the command line (for example, `mkuser id=234 foo`, and ID 234 is already taken by a user in any of the registries).

uniqbyname
This value checks 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 for a `mkuser id=123 foo` type of command. If the ID is not specified from the command line, the new account might not have the same ID value as an existing account with the same name in another registry. For example, *acct1* with ID 234 is a local account. When creating an LDAP account *acct1*, `mkuser -R LDAP acct1` might pick a unique ID of 235 for the LDAP account. The result is *acct1* with ID 234 on local, and *acct1* with 235 on LDAP.

Note: ID collision detection in the target registry is always enforced regardless of the **dist_uniqid** attribute.

The **uniqbyname** value works well against two registries. With more than two registries, and when ID collision already exists between two registries, the behavior of **mkuser** (**mkgroup**) is unspecified when creating a new account in a third registry using the colliding ID values. The new account creation might succeed or fail depending the order the registries are checked.

For example: Suppose a system is configured with three registries: local, LDAP and DCE. An *acct1* account exists in LDAP and an *acct2* account in DCE, both with ID 234. When the system administrator executes the `mkuser -R files id=234 acct1` (`mkgroup -R files id=234 acct1`) command to create the local account with the **uniqbyname** value, the **mkuser** (**mkgroup**) command checks against the LDAP registry first, and finds that ID 234 is taken by LDAP account *acct1*. Since the account to be created has the same account name, the **mkuser** (**mkgroup**) command successfully creates the local account *acct1* with ID 234. If the DCE registry is checked first, the **mkuser** (**mkgroup**) command finds that ID 234 is taken by DCE account *acct2*, and creation of local account *acct1* fails. The check for ID collision enforces ID uniqueness between the local registry and remote registries or between remote registries. There is no guarantee of ID uniqueness between the newly created account on the remote registry and existing local users on other systems which use the same remote registry. The **mkuser** (**mkgroup**) command bypasses the remote registry if it is not reachable at the time the command is run.

Root account

The root account has virtually unlimited access to all programs, files, and resources on a system.

The root account is the special user in the **/etc/passwd** file with the user ID (UID) of `0` and is commonly given the user name, *root*. It is not the user name that makes the root account so special, but the UID value of `0`. This means that any user that has a UID of `0` also has the same privileges as the root user. Also, the root account is always authenticated by means of the local security files.

The root account should always have a password, which should never be shared. The root account should be given a password immediately after the system is installed. Only the system administrator should know the root password. System administrators should only operate as the root user to perform system administration functions that require root privileges. For all other operations, they should return to their normal user account.

Attention: Routinely operating as the root user can result in damage to the system because the root account overrides many safeguards in the system.

Disabling direct root login

A common attack method of potential hackers is to obtain the root password.

To avoid this type of attack, you can disable direct access to your root ID and then require your system administrators to obtain root privileges by using the **su -** command. In addition to allowing you to remove the root user as a point of attack, restricting direct root access allows you to monitor which users gained root access, as well as the time of their action. You can do this by viewing the **/var/adm/sulog** file. Another alternative is to enable system auditing, which will report this type of activity.

To disable remote login access for your root user, edit the **/etc/security/user** file. Specify False as the rlogin value on the entry for root.

Before you disable the remote root login, examine and plan for situations that would prevent a system administrator from logging in under a non-root user ID. For example, if a user's home file system is full, the user would not be able to log in. If the remote root login were disabled and the user who could use the **su -** command to change to root had a full home file system, root could never take control of the system. This issue can be bypassed by system administrators creating home file systems for themselves that are larger than the average user's file system.

For more information about controlling root login, see "CAPP/EAL4+ system configuration" on page 11.

Administrative roles

You can assign portions of root-user authority to non-root users. Different root-user tasks are assigned different authorizations. These authorizations are grouped into roles and assigned to different users.

This section covers the following topics:

- “Roles overview”
- “Setting up and maintaining roles using SMIT”
- “Authorizations overview” on page 27

Roles overview

Roles consist of authorizations that allow a user to run functions that normally would require root-user permission.

The following is a list of valid roles:

Add and Remove Users
Allows any user to act as the root user for this role. They are able to add and remove users, change information about a user, modify audit classes, manage groups, and change passwords. Anyone who performs user administration must be in the **security** group.

Change Users Password
Allows a user to change a passwords.

Manage Roles
Allows a user to create, change, remove and list roles. The user must be in the **security** group.

Backup and Restore
Allows a user to back up and restore file systems and directories. This role is not sufficient to enable a system backup and restore using **mksysb** and requires proper authorizations.

Backup Only
Allows a user only to back up file systems and directories. The user must have the proper authorization to enable a system backup.

Run Diagnostics
Allows a user or service representative to run diagnostics and diagnostic tasks. The user must have **system** specified as the primary group and also a group set that includes **shutdown**.
Note: Users in the Run Diagnostics role can change the system configuration, update microcode, and so on. Users in this role must fully understand the responsibility that the role requires.

System Shutdown
Allows a user to shut down, reboot, or halt a system. Anyone who performs this role must have a group set that includes **shutdown**.

Setting up and maintaining roles using SMIT

You can use SMIT fast paths to add, change, display, remove or list roles.

The following SMIT fast paths are available for implementing and maintaining roles:

Table 4. Setting Up and Maintaining Roles Tasks

Task	SMIT Fast Path
Add a Role	**smit mkrole**
Change Characteristics of a Role	**smit chrole**
Show Characteristics of a Role	**smit lsrole**

Table 4. Setting Up and Maintaining Roles Tasks (continued)

Task	SMIT Fast Path
Remove a Role	**smit rmrole**
List All Roles	**smit lsrole**

Authorizations overview

Authorizations are authority attributes for a user. These authorizations allow a user to do certain tasks.

The following types of authorization exist:

Primary Authorization
Allows a user to run a specific command. For example, RoleAdmin authorization is a primary authorization allowing a user administrator to run the **chrole** command. Without this authorization, the command terminates without modifying the role definitions.

Authorization modifier
Increases the capability of a user. For example, UserAdmin authorization is an authorization modifier that increases the capability of a user administrator belonging to the security group. Without this authorization, the **mkuser** command only creates non-administrative users. With this authorization, the **mkuser** command also creates administrative users.

The authorizations perform the following functions:

Backup
Performs a system backup. The following command uses the Backup authorization:

Backup
Backs up files and file systems. The user administrator must have Backup authorization.

Diagnostics
Allows a user to run diagnostics. This authority is also required to run diagnostic tasks directly from the command line. The following command uses the Diagnostics authorization:

diag Runs diagnostics on selected resources. If the user administrator does not have Diagnostics authority, the command ends.

GroupAdmin
Performs the functions of the root user on group data. The following commands use the GroupAdmin authorization:

chgroup
Changes any group information. If the user does not have GroupAdmin authorization, they can only change non-administrative group information.

chgrpmem
Administers all groups. If the group administrator does not have GroupAdmin authorization, they can only change the membership of the group they administer or a user in group security to administer any non-administrative group.

chsec Modifies administrative group data in the **/etc/group** and **/etc/security/group** files. The user can also modify the default stanza values. If the user does not have GroupAdmin authorization, they can only modify non-administrative group data in the **/etc/group** and **/etc/security/group** files.

mkgroup
Creates any group. If the user does not have GroupAdmin authorization, the user can only create non-administrative groups.

rmgroup
Removes any group. If the user does not have GroupAdmin authorization, the user can only remove non-administrative groups.

ListAuditClasses
Views the list of valid audit classes. The user administrator who uses this authorization does not have to be the root user or in the audit group.

Use the **smit mkuser** or **smit chuser** fast path to list audit classes available to make or change a user. Enter the list of audit classes in the **AUDIT classes** field.

PasswdAdmin
Performs the functions of the root user on password data. The following commands use the PasswdAdmin authorization:

chsec Modifies the **lastupdate** and **flags** attributes of all users. Without the PasswdAdmin authorization, the **chsec** command allows the user administrator to modify only the **lastupdate** and **flags** attribute of non-administrative users.

lssec Views the **lastupdate** and **flags** attributes of all users. Without the PasswdAdmin authorization, the **lssec** command allows the user administrator to only view the **lastupdate** and **flags** attribute of non-administrative users.

pwdadm
Changes the password of all users. The user administrator must be in the security group.

PasswdManage
Performs password administration functions on non-administrative users. The following command uses the PasswdManage authorization:

pwdadm
Changes the password of a non-administrative user. The administrator must be in the security group or have the PasswdManage authorization.

UserAdmin
Performs the functions of the root user on user data. Only users with UserAdmin authorization can modify the role information of a user. You cannot access or modify user auditing information with this authorization. The following commands use the UserAdmin authorization:

chfn Changes any user general information (gecos) field. If the user does not have UserAdmin authorization but is in the security group, they can change any non-administrative user gecos field. Otherwise, users can only change their own gecos field.

chsec Modifies administrative user data in the **/etc/passwd**, **/etc/security/environ**, **/etc/security/lastlog**, **/etc/security/limits**, and **/etc/security/user** files, including the roles attribute. The user administrator can also modify the default stanza values and the **/usr/lib/security/mkuser.default** file, excluding the auditclasses attributes.

chuser
Changes any user's information except for the auditclasses attribute. If the user does not have UserAdmin authorization, they can only change non-administrative user information, except for the auditclasses and roles attributes.

mkuser
Creates any user, except for the auditclasses attribute. If the user does not have UserAdmin authorization, the user can only create non-administrative users, except for the auditclasses and roles attributes.

rmuser
Removes any user. If the user administrator does not have UserAdmin authorization, they can only create non-administrative users.

UserAudit

Allows the user to modify user-auditing information. The following commands use the UserAudit authorization:

chsec Modifies the auditclasses attribute of the **mkuser.default** file for non-administrative users. If the user has UserAdmin authorization, they can also modify the auditclasses attribute of the **mkuser.default** file for administrative and non-administrative users.

chuser
Modifies the auditclasses attribute of a non-administrative user. If the user administrator has UserAdmin authorization, they can also modify the auditclasses attribute of all users.

lsuser Views the auditclasses attribute of a non-administrative user if the user is root user or in the security group. If the user has UserAdmin authorization, they can also view the auditclasses attribute of all users.

mkuser
Creates a new user and allows user administrator to assign the auditclasses attribute of a non-administrative user. If the user has UserAdmin authorization, they can also modify the auditclasses attribute of all users.

RoleAdmin

Performs the functions of the root user on role data. The following commands use the RoleAdmin authorization:

chrole Modifies a role. If the user administrator does not have the RoleAdmin authorization, the command ends.

lsrole Views a role.

mkrole
Creates a role. If the user administrator does not have the RoleAdmin authorization, the command ends.

rmrole
Removes a role. If the user administrator does not have the RoleAdmin authorization, the command ends.

Restore

Performs a system restoration. The following command uses the Restore authorization:

Restore
Restores backed-up files. The user administrator must have Restore authorization.

Authorization commands list:

Authorization commands list with permissions and authorizations.

The following table lists the commands and the authorizations they use.

Command	Permissions	Authorizations
chfn	2555 root.security	UserAdmin
chuser	4550 root.security	UserAdmin, UserAudit
diag	0550 root.system	Diagnostics
lsuser	4555 root.security	UserAudit, UserAdmin
mkuser	4550 root.security	UserAdmin, UserAudit
rmuser	4550 root.security	UserAdmin
chgroup	4550 root.security	GroupAdmin
lsgroup	0555 root.security	GroupAdmin

Command	Permissions	Authorizations
mkgroup	4550 root.security	GroupAdmin
rmgroup	4550 root.security	GroupAdmin
chgrpmem	2555 root.security	GroupAdmin
pwdadm	4555 root.security	PasswdManage, PasswdAdmin
passwd	4555 root.security	PasswdManage, PasswdAdmin
chsec	4550 root.security	UserAdmin, GroupAdmin, PasswdAdmin, UserAudit
lssec	0550 root.security	PasswdAdmin
chrole	4550 root.security	RoleAdmin
lsrole	0550 root.security	RoleAdmin
mkrole	4550 root.security	RoleAdmin
rmrole	4550 root.security	RoleAdmin
backup	4555 root.system	Backup
restore	4555 root.system	Restore

User accounts

This section discusses security administrative tasks for user accounts.

Recommended user attributes

User administration consists of creating users and groups and defining their attributes.

A major attribute of users is how they are authenticated. Users are the primary agents on the system. Their attributes control their access rights, environment, how they are authenticated, as well as how, when, and where their accounts can be accessed.

Groups are collections of users who can share the same access permissions for protected resources. A group has an ID and is composed of members and administrators. The creator of the group is usually the first administrator.

Many attributes can be set for each user account, including password and login attributes. For a list of configurable attributes, refer to “Disk quota system overview” on page 44. The following attributes are recommended:

- Each user should have a user ID that is not shared with any other user. All of the security safeguards and accountability tools work only if each user has a unique ID.
- Give user names that are meaningful to the users on the system. Actual names are best, because most electronic mail systems use the user ID to label incoming mail.
- Add, change, and delete users using the Web-based System Manager or SMIT interface. Although you can perform all of these tasks from the command line, these interfaces help reduce small errors.
- Do not give an initial password to a user account until the user is ready to log in to the system. If the password field is defined as an * (asterisk) in the **/etc/passwd** file, account information is kept, but no one can log in to that account.
- Do not change the system-defined user IDs that are needed by the system to function correctly. The system-defined user IDs are listed in the **/etc/passwd** file.
- In general, do not set the *admin* parameter to `true` for any user IDs. Only the root user can change attributes for users with `admin=true` set in the **/etc/security/user** file.

The operating system supports the standard user attributes usually found in the **/etc/passwd** and **/etc/system/group** files, such as:

Authentication Information Specifies the password

Credentials Specifies the user identifier, principal group, and the supplementary group ID

Environment Specifies the home or shell environment.

User account control

User accounts have attributes that can be altered.

Each user account has a set of associated attributes. These attributes are created from default values when a user is created by using the **mkuser** command. The attributes can be altered by using the **chuser** command. The following are the user attributes that control login and are not related to password quality:

account_locked If an account must be explicitly locked, this attribute can be set to True; the default is False.

admin If set to True, this user can not change the password. Only the administrator can change it.

admgroups Lists groups for which this user has administrative rights. For those groups, the user can add or delete members.

auth1 The authentication method that is used to grant the user access. Typically, it is set to `SYSTEM`, which will then use newer methods.

auth2 Method that runs after the user has been authenticated by whatever was specified in **auth1**. It cannot block access to the system. Typically, it is set to `NONE`.

daemon This boolean parameter specifies whether the user is allowed to start daemons or subsystems with the **startsrc** command. It also restricts the use of the cron and at facilities.

login Specifies whether this user is allowed to log in. A successful login resets the **unsuccessful_login_count** attribute to a value of `0` (from the **loginsuccess** subroutine).

logintimes Restricts when a user can log in. For example, a user might be restricted to accessing the system only during normal business hours.

registry Specifies the user registry. It can be used to tell the system about alternate registries for user information, such as NIS, LDAP, or Kerberos.

rlogin Specifies whether this user is allowed to log in by using **rlogin** or **telnet**.

su Specifies whether other users can switch to this ID with the **su** command.

sugroups Specifies which groups are allowed to switch to this user ID.

ttys Limits certain accounts to physically secure areas.

expires Manages student or guest accounts; also can be used to turn off accounts temporarily.

loginretries Specifies the maximum number of consecutive failed login attempts before the user ID is locked by the system. The failed attempts are recorded in the **/etc/security/lastlog** file.

umask Specifies the initial **umask** for the user.

rcmds Specifies whether the user account can be accessed with the **rsh** or **exec** commands. A value of `allow` indicates that the account may be accessed by **rsh** and **rexec**. A value of `deny` indicates no account access by **rsh** and **rexec** commands. A value of `hostlogincontrol` indicates that the account access is controlled by **hostallowedlogin** and **hostsdeniedlogin** attributes.

hostallowedlogin Specifies the hosts which permit the user to login. This attribute is intended to be used in a networked environment where user attributes are shared by multiple hosts.

hostsdeniedlogin Specifies the hosts which do not permit the user to login. This attribute is intended to be used in a networked environment where user attributes are shared by multiple hosts.

maxulogs Specifies the maximum number of logins per user. If the user has reached the maximum number of allowed logins, login will be denied.

The complete set of user attributes is defined in the **/etc/security/user**, **/etc/security/limits**, **/etc/security/audit/config** and **/etc/security/lastlog** files. The default for user creation with the **mkuser** command is specified in the **/usr/lib/security/mkuser.default** file. Only options that override the general defaults in the default stanzas of the **/etc/security/user** and **/etc/security/limits** files, as well as audit classes, must be specified in the **mkuser.default** file. Several of these attributes control how a user can log in, and they can be configured to lock the user account (prevent further logins) automatically under specified conditions.

After the user account has been locked by the system due to the number of unsuccessful login attempts, the user is not able to log in until the system administrator resets the user **unsuccessful_login_count** attribute in the **/etc/security/lastlog** file to be less than the value of login retries. This can be done using the following **chsec** command, as follows:

```
chsec -f /etc/security/lastlog -s username -a
unsuccessful_login_count=0
```

The defaults can be changed by using the **chsec** command to edit the default stanza in the appropriate security file, such as the **/etc/security/user** or **/etc/security/limits** files. Many of the defaults are defined to be the standard behavior. To explicitly specify attributes that are set every time that a new user is created, change the *user* entry in **/usr/lib/security/mkuser.default**.

For information on extended user password attributes, refer to "Passwords" on page 38.

Login user IDs

The operating system identifies users by their login user ID.

The login user ID allows the system to trace all user actions to their source. After a user logs in to the system but before running the initial user program, the system sets the login ID of the process to the user ID found in the user database. All subsequent processes during the login session are tagged with this ID. These tags provide a trail of all activities performed by the login user ID. The user can reset the effective user ID, real user ID, effective group ID, real group ID, and supplementary group ID during the session, but cannot change the login user ID.

Strengthening user security with Access Control Lists

To achieve an appropriate level of security in your system, develop a consistent security policy to manage user accounts. The most commonly used security mechanism is the access control list (ACL).

For information about ACLs and developing a security policy, see "Access Control Lists" on page 46.

PATH environment variable

The **PATH** environment variable is an important security control. It specifies the directories to be searched to find a command.

The default systemwide **PATH** value is specified in the **/etc/profile** file, and each user normally has a **PATH** value in the user's **$HOME/.profile** file. The **PATH** value in the **.profile** file either overrides the systemwide **PATH** value or adds extra directories to it.

Unauthorized changes to the **PATH** environment variable can enable a user on the system to ″spoof″ other users (including root users). *Spoofing* programs (also called *Trojan horse* programs) replace system commands and then capture information meant for that command, such as user passwords.

For example, suppose a user changes the **PATH** value so that the system searches the **/tmp** directory first when a command is run. Then the user places in the **/tmp** directory a program called **su** that asks for the root password just like the **su** command. Then the **/tmp/su** program mails the root password to the user and calls the real **su** command before exiting. In this scenario, any root user who used the **su** command would reveal the root password and not even be aware of it.

To prevent any problems with the **PATH** environment variable for system administrators and users, do the following:

- When in doubt, specify full path names. If a full path name is specified, the **PATH** environment variable is ignored.
- Never put the current directory (specified by **.** (period)) in the **PATH** value specified for the root user. Never allow the current directory to be specified in **/etc/profile**.

- The root user should have its own **PATH** specification in his private **.profile** file. Typically, the specification in **/etc/profile** lists the minimal standard for all users, whereas the root user might need more or fewer directories than the default.
- Warn other users not to change their **.profile** files without consulting the system administrator. Otherwise, an unsuspecting user could make changes that allow unintended access. A user **.profile** file should have permissions set to 740.
- System administrators should not use the **su** command to gain root privilege from a user session, because the user's **PATH** value specified in the **.profile** file is in effect. Users can set their own **.profile** files. System administrators should log in to the user's machine as root user or preferably, using their own ID and then use the following command:

  ```
  /usr/bin/su - root
  ```

  This ensures that the root environment is used during the session. If a system administrator does operate as root in another user session, the system administrator should specify full path names throughout the session.
- Protect the input field separator (**IFS**) environment variable from being changed in the **/etc/profile** file. The **IFS** environment variable in the **.profile** file can be used to alter the **PATH** value.

Anonymous FTP with a secure user account setup

Setting up anonymous FTP with a secure user account.

Things to Consider

> The information in this how-to was tested using AIX 5.2. If you are using a different version or level of AIX, the results you obtain might vary significantly.

This scenario sets up an anonymous FTP with a secure user account, using the command line interface and a script.

Note: This scenario cannot be used on a system with the Controlled Access Protection Profile (CAPP) with Evaluation Assurance Level 4+ (EAL4+) feature.

1. Verify that the **bos.net.tcp.client** fileset is installed on your system, by typing the following command:

   ```
   lslpp -L | grep bos.net.tcp.client
   ```

   If you receive no output, the fileset is not installed. For instructions on how to install it, see the *AIX 5L Version 5.3 Installation Guide and Reference*.
2. Verify that you have at least 8 MB of free space available in the system's **/home** directory, by typing the following command:

   ```
   df -k /home
   ```

   The script in step 4 requires at least 8 MB free space in the **/home** directory to install the required files and directories. If you need to increase the amount of available space, see the *AIX 5L Version 5.3 System Management Guide: Operating System and Devices*.
3. With root authority, change to the **/usr/samples/tcpip** directory. For example:

   ```
   cd /usr/samples/tcpip
   ```
4. To set up the account, run the following script:

   ```
   ./anon.ftp
   ```
5. When prompted with `Are you sure you want to modify /home/ftp?`, type `yes`. Output similar to the following displays:

   ```
   Added user anonymous.
   Made /home/ftp/bin directory.
   Made /home/ftp/etc directory.
   ```

```
Made /home/ftp/pub directory.
Made /home/ftp/lib directory.
Made /home/ftp/dev/null entry.
Made /home/ftp/usr/lpp/msg/en_US directory.
```

6. Change to the **/home/ftp** directory. For example:

   ```
   cd /home/ftp
   ```

7. Create a **home** subdirectory, by typing:

   ```
   mkdir home
   ```

8. Change the permissions of the **/home/ftp/home** directory to `drwxr-xr-x`, by typing:

   ```
   chmod 755 home
   ```

9. Change to the **/home/ftp/etc** directory, by typing:

   ```
   cd /home/ftp/etc
   ```

10. Create the **objrepos** subdirectory, by typing:

    ```
    mkdir objrepos
    ```

11. Change the permissions of the **/home/ftp/etc/objrepos** directory to `drwxrwxr-x`, by typing:

    ```
    chmod 775 objrepos
    ```

12. Change the owner and group of the **/home/ftp/etc/objrepos** directory to the root user and the system group, by typing:

    ```
    chown root:system objrepos
    ```

13. Create a **security** subdirectory, by typing

    ```
    mkdir security
    ```

14. Change the permissions of the **/home/ftp/etc/security** directory to `drwxr-x---`, by typing:

    ```
    chmod 750 security
    ```

15. Change the owner and group of the **/home/ftp/etc/security** directory to the root user and the security group, by typing:

    ```
    chown root:security security
    ```

16. Change to the **/home/ftp/etc/security** directory, by typing:

    ```
    cd security
    ```

17. Add a user by typing the following SMIT fast path:

    ```
    smit mkuser
    ```

    In this scenario, we are adding a user named `test`.

18. In the SMIT fields, enter the following values:

    ```
    User NAME                                           [test]
    ADMINISTRATIVE USER?                                 true
    Primary GROUP                                       [staff]
    Group SET                                           [staff]
    Another user can SU TO USER?                         true
    HOME directory                                      [/home/test]
    ```

    After you enter your changes, press Enter to create the user. After the SMIT process completes, exit SMIT.

19. Create a password for this user with the following command:

    ```
    passwd test
    ```

    When prompted, enter the desired password. You must enter the new password a second time for confirmation.

20. Change to the **/home/ftp/etc** directory, by typing

    ```
    cd /home/ftp/etc
    ```

21. Copy the **/etc/passwd** file to the **/home/ftp/etc/passwd** file, using the following command:

    ```
    cp /etc/passwd /home/ftp/etc/passwd
    ```

22. Using your favorite editor, edit the **/home/ftp/etc/passwd** file. For example:

    ```
    vi passwd
    ```

23. Remove all lines from the copied content except those for the root, ftp, and test users. After your edit, the content should look similar to the following:

    ```
    root:!:0:0::/:/bin/ksh
    ftp:*:226:1::/home/ftp:/usr/bin/ksh
    test:!:228:1::/home/test:/usr/bin/ksh
    ```

24. Save your changes and exit the editor.
25. Change the permissions of the **/home/ftp/etc/passwd** file to `-rw-r--r--`, by typing:

    ```
    chmod 644 passwd
    ```

26. Change the owner and group of the **/home/ftp/etc/passwd** file to the root user and the security group, by typing:

    ```
    chown root:security passwd
    ```

27. Copy the contents of the **/etc/security/passwd** file to the **/home/ftp/etc/security/passwd** file, using the following command:

    ```
    cp /etc/security/passwd /home/ftp/etc/security/passwd
    ```

28. Using your favorite editor, edit the **/home/ftp/etc/security/passwd** file. For example:

    ```
    vi ./security/passwd
    ```

29. Remove all stanzas from the copied content except the stanza for the test user.
30. Remove the `flags = ADMCHG` line from the test user stanza. After your edits, the content should look similar to the following:

    ```
    test:
            password = 2HaAYgpDZX3Tw
            lastupdate = 990633278
    ```

31. Save your changes and exit the editor.
32. Change the permissions of the **/home/ftp/etc/security/passwd** file to `-rw-------`, by typing:

    ```
    chmod 600 ./security/passwd
    ```

33. Change the owner and group of the **/home/ftp/etc/security/passwd** file to the root user and the security group, by typing:

    ```
    chown root:security ./security/passwd
    ```

34. Using your favorite editor, edit the **/home/ftp/etc/security/group** file. For example:

    ```
    vi ./security/group
    ```

35. Add the following lines to the file:

    ```
    system:*:0:
    staff:*:1:test
    ```

36. Save your changes and exit the editor.
37. Use the following commands to copy the appropriate content into the **/home/ftp/etc/objrepos** directory:

    ```
    cp /etc/objrepos/CuAt ./objrepos
    cp /etc/objrepos/CuAt.vc ./objrepos
    cp /etc/objrepos/CuDep ./objrepos
    cp /etc/objrepos/CuDv ./objrepos
    cp /etc/objrepos/CuDvDr ./objrepos
    cp /etc/objrepos/CuVPD ./objrepos
    cp /etc/objrepos/Pd* ./objrepos
    ```

38. Change to the **/home/ftp/home** directory, by typing:

    ```
    cd ../home
    ```

39. Make a new home directory for your user, by typing:

    ```
    mkdir test
    ```

    This will be the home directory for the new ftp user.

40. Change the owner and group of the **/home/ftp/home/test** directory to the `test` user and the staff group, by typing:

    ```
    chown test:staff test
    ```

41. Change the permissions of the **/home/ftp/home/test** file to `-rwx------`, by typing:

    ```
    chmod 700 test
    ```

At this point, you have ftp sublogin set up on your machine. You can test this with the following procedure:

1. Using ftp, connect to the host on which you created the `test` user. For example:

   ```
   ftp MyHost
   ```

2. Log in as `anonymous`. When prompted for a password, press Enter.
3. Switch to the newly created `test` user, by using the following command:

   ```
   user test
   ```

   When prompted for a password, use the password you created in step 19 on page 34
4. Use the pwd command to verify the user's home directory exists. For example:

   ```
   ftp> pwd
        /home/test
   ```

   The output shows **/home/test** as an **ftp** subdirectory. The full path name on the host is actually **/home/ftp/home/test**.

For more information:

- ″TCP/IP Security″ in *AIX 5L Version 5.3 Security Guide*
- ″**ftp Command**″ in *AIX 5L Version 5.3 Commands Reference*

System special user accounts

AIX provides a default set of system special user accounts that prevents the root and system accounts from owning all operating system files and file systems.

Attention: Use caution when removing a system special user account. You can disable a specific account by inserting an asterisk (*) at the beginning of its corresponding line of the **/etc/security/passwd** file. However, be careful not to disable the root user account. If you remove system special user accounts or disable the root account, the operating system will not function.

The following accounts are predefined in the operating system:

adm The adm user account owns the following basic system functions:

- Diagnostics, the tools for which are stored in the **/usr/sbin/perf/diag_tool** directory.
- Accounting, the tools for which are stored in the following directories:
  - **/usr/sbin/acct**
  - **/usr/lib/acct**
  - **/var/adm**
  - **/var/adm/acct/fiscal**
  - **/var/adm/acct/nite**
  - **/var/adm/acct/sum**

bin The bin user account typically owns the executable files for most user commands. This account's primary purpose is to help distribute the ownership of important system directories and files so that everything is not owned solely by the root and sys user accounts.

daemon
The daemon user account exists only to own and run system server processes and their associated files. This account guarantees that such processes run with the appropriate file access permissions.

nobody
The nobody user account is used by the Network File System (NFS) to enable remote printing. This account exists so that a program can permit temporary root access to root users. For example, before enabling Secure RPC or Secure NFS, check the **/etc/public** key on the master NIS server to find a user who has not been assigned a public key and a secret key. As root user, you can create an entry in the database for each unassigned user by entering:

```
newkey -u username
```

Or, you can create an entry in the database for the nobody user account, and then any user can run the **chkey** program to create their own entries in the database without logging in as root.

root
The root user account, UID 0, through which you can perform system maintenance tasks and troubleshoot system problems.

sys
The sys user owns the default mounting point for the Distributed File Service (DFS™) cache, which must exist before you can install or configure DFS on a client. The **/usr/sys** directory can also store installation images.

system
System group is a system-defined group for system administrators. Users of the system group have the privilege to perform some system maintenance tasks without requiring root authority.

Removing unnecessary default user accounts

During installation of the operating system, a number of default user and group IDs are created. Depending on the applications you are running on your system and where your system is located in the network, some of these user and group IDs can become security weaknesses, vulnerable to exploitation. If these users and group IDs are not needed, you can remove them to minimize security risks associated with them.

The following table lists the most common default user IDs that you might be able to remove:

Table 5. Common default user IDs that you might be able to remove.

User ID	Description
uucp, nuucp	Owner of hidden files used by uucp protocol. The uucp user account is used for the UNIX-to-UNIX Copy Program, which is a group of commands, programs, and files, present on most AIX systems, that allows the user to communicate with another AIX system over a dedicated line or a telephone line.
lpd	Owner of files used by printing subsystem
guest	Allows access to users who do not have access to accounts

The following table lists common group IDs that might not be needed:

Table 6. Common group IDs that might not be needed.

Group ID	Description
uucp	Group to which uucp and nuucp users belong
printq	Group to which lpd user belongs

Analyze your system to determine which IDs are indeed not needed. There might also be additional user and group IDs that you might not need. Before your system goes into production, perform a thorough evaluation of available IDs.

Passwords

Guessing passwords is one of the most common attack methods that a system experiences. Therefore, controlling and monitoring your password-restriction policy is essential.

AIX provides mechanisms to help you enforce a stronger password policy, such as establishing values for the following:

- Minimum and maximum number of weeks that can elapse before and after a password can be changed
- Minimum length of a password
- Minimum number of alphabetic characters that can be used when selecting a password

Establishing good passwords

Good passwords are effective first lines of defense against unauthorized entry into a system.

Passwords are effective if they are:

- A mixture of both uppercase and lowercase letters
- A combination of alphabetic, numeric, or punctuation characters. Also, they may have special characters such as `~!@#$%^&*()-_=+[]{}|\;:'",.<>?/<space>`
- Are not written down anywhere
- Are at least 7 to a maximum of 8 characters in length, if using the **/etc/security/passwd** file (authentication implementations that use registries, such as LDAP, can have passwords that exceed this maximum length)
- Are not real words that can be found in any dictionary
- Are not patterns of letters on the keyboard, like *qwerty*
- Are not real words or known patterns spelled backwards
- Do not contain any personal information about yourself, family, or friends
- Do not follow the same pattern as a previous password
- Can be typed relatively quickly so someone nearby cannot determine your password

In addition to these mechanisms, you can further enforce stricter rules by restricting passwords so that they cannot include standard UNIX words, which can be guessed. This feature uses the dictionlist, which requires that you first have the **bos.data** and **bos.txt** file sets installed.

To implement the previously defined dictionlist, edit the following line in the **/etc/security/users** file:

```
dictionlist = /usr/share/dict/words
```

The **/usr/share/dict/words** file uses the dictionlist to prevent standard UNIX words from being used as passwords.

Using the /etc/passwd file

Traditionally, the **/etc/passwd** file is used to keep track of every registered user that has access to a system.

The **/etc/passwd** file is a colon-separated file that contains the following information:

- User name
- Encrypted password
- User ID number (UID)
- User's group ID number (GID)

- Full name of the user (GECOS)
- User home directory
- Login shell

The following is an example of an **/etc/passwd** file:

```
root:!:0:0::/:/usr/bin/ksh
daemon:!:1:1::/etc:
bin:!:2:2::/bin:
sys:!:3:3::/usr/sys:
adm:!:4:4::/var/adm:
uucp:!:5:5::/usr/lib/uucp:
guest:!:100:100::/home/guest:
nobody:!:4294967294:4294967294::/:
lpd:!:9:4294967294::/:
lp:*:11:11::/var/spool/lp:/bin/false
invscout:*:200:1::/var/adm/invscout:/usr/bin/ksh
nuucp:*:6:5:uucp login user:/var/spool/uucppublic:/usr/sbin/uucp/uucico
paul:!:201:1::/home/paul:/usr/bin/ksh
jdoe:*:202:1:John Doe:/home/jdoe:/usr/bin/ksh
```

AIX does not store encrypted passwords in the **/etc/password** file in the way that UNIX systems do, but in the[1] file by default, which is only readable by the root user. The password filed in **/etc/passwd** is used by AIX to signify if there is a password or whether the account is blocked.

The **/etc/passwd** file is owned by the root user and must be readable by all the users, but only the root user has writable permissions, which is shown as `-rw-r--r--`. If a user ID has a password, then the password field will have an `!` (exclamation point). If the user ID does not have a password, then the password field will have an `*` (asterisk). The encrypted passwords are stored in the **/etc/security/passwd** file. The following example contains the last four entries in the **/etc/security/passwd** file based on the entries from the **/etc/passwd** file shown previously.

```
guest:
        password = *

nobody:
        password = *

lpd:
        password = *

paul:
        password = eacVScDKri4s6
        lastupdate = 1026394230
        flags = ADMCHG
```

The user ID `jdoe` does not have an entry in the **/etc/security/passwd** file because it does not have a password set in the **/etc/passwd** file.

The consistency of the **/etc/passwd** file can be checked using the **pwdck** command. The **pwdck** command verifies the correctness of the password information in the user database files by checking the definitions for all of the users or for specified users.

Using the /etc/passwd file and network environments

In a traditional networked environment, a user must have had an account on each system to gain access to that system.

That typically meant that the user would have an entry in each of the **/etc/passwd** files on each system. However, in a distributed environment, there is no easy way to ensure that every system had the same

[1] /etc/security/password

/etc/passwd file. To solve this problem, several methods make the information in the **/etc/passwd** file available over the network, including Network Information System (NIS) and NIS+.

For more information about NIS and NIS+, see "Network Information Services and NIS+ security" on page 199.

Hiding user names and passwords

To achieve a higher level of security, ensure that user IDs and passwords are not visible within the system.

The **.netrc** files contain user IDs and passwords. This file is not protected by encryption or encoding, thus its contents are clearly shown as plain text. To find these files, run the following command:

```
# find `awk -F: '{print $6}' /etc/passwd` -name .netrc -ls
```

After you locate these files, delete them. A more effective way to save passwords is by setting up Kerberos. For more information about Kerberos, see "Kerberos" on page 217.

Setting recommended password options

Proper password management can only be accomplished through user education. To provide some additional security, the operating system provides configurable password restrictions. These allow the administrator to constrain the passwords chosen by users and to force passwords to be changed regularly.

Password options and extended user attributes are located in the **/etc/security/user** file, an ASCII file that contains attribute stanzas for users. These restrictions are enforced whenever a new password is defined for a user. All password restrictions are defined per user. By keeping restrictions in the default stanza of the **/etc/security/user** file, the same restrictions are enforced on all users. To maintain password security, all passwords must be similarly protected.

Administrators can also extend the password restrictions. Using the **pwdchecks** attribute of the **/etc/security/user** file, an administrator can add new subroutines (known as *methods*) to the password restrictions code. Thus, local site policies can be added to and enforced by the operating system. For more information, see "Extending password restrictions" on page 42.

Apply password restrictions sensibly. Attempts to be too restrictive, such as limiting the password space, which makes guessing the password easier, or forcing the user to select passwords that are difficult to remember, which might then be written down, can jeopardize password security. Ultimately, password security rests with the user. Simple password restrictions, coupled with sensible guidelines and an occasional audit to verify that current passwords are unique, are the best policy.

The following table lists recommended values for some security attributes related to user passwords in the **/etc/security/user** file.

Table 7. Recommended security attribute values for user passwords.

Attribute	Description	Recommended Value	Default Value	Maximum Value
dictionlist	Verifies passwords do not include standard UNIX words.	**/usr/share/dict/words**	Not applicable	Not applicable
histexpire	Number of weeks before password can be reused.	26	0	260*
histsize	Number of password iterations allowed.	20	0	50

Table 7. Recommended security attribute values for user passwords. (continued)

Attribute	Description	Recommended Value	Default Value	Maximum Value
maxage	Maximum number of weeks before password must be changed.	8	0	52
maxexpired	Maximum number of weeks beyond *maxage* that an expired password can be changed by the user. (Root is exempt.)	2	-1	52
maxrepeats	Maximum number of characters that can be repeated in passwords.	2	8	8
minage	Minimum number of weeks before a password can be changed. This should not be set to a nonzero value unless administrators are always easy to reach to reset an accidentally compromised password that was recently changed.	0	0	52
minalpha	Minimum number of alphabetic characters required on passwords.	2	0	8
mindiff	Minimum number of unique characters that passwords must contain.	4	0	8
minlen	Minimum length of password.	6 (8 for root user)	0	8
minother	Minimum number of non-alphabetic characters required on passwords.	2	0	8
pwdwarntime	Number of days before the system issues a warning that a password change is required.	5	Not applicable	Not applicable

Table 7. Recommended security attribute values for user passwords. (continued)

Attribute	Description	Recommended Value	Default Value	Maximum Value
pwdchecks	This entry can be used to augment the **passwd** command with a custom code that checks the password quality.	For more information. see "Extending password restrictions."	Not applicable	Not applicable

* A maximum of 50 passwords are retained.

For a Controlled Access Protection Profile and Evaluation Assurance Level 4+ (CAPP/EAL4+) system, use the values recommended in "User and port configuration" on page 12.

If text processing is installed on the system, the administrator can use the **/usr/share/dict/words** file as a **dictionlist** dictionary file. In such a case, the administrator can set the **minother** attribute to `0`. Because most words in the dictionary file do not contain characters that fall into the **minother** attribute category, setting the **minother** attribute to `1` or more eliminates the need for the vast majority of words in this dictionary file.

The minimum length of a password on the system is set by the value of the **minlen** attribute or the value of the **minalpha** attribute plus the value of the **minother** attribute, whichever is greater. The maximum length of a password is 8 characters. The value of the **minalpha** attribute plus the value of the **minother** attribute must never be greater than 8. If the value of the **minalpha** plus the value of the **minother** attribute is greater than 8, the value of the **minother** attribute is reduced to 8 minus the value of the **minalpha** attribute.

If the values of both the **histexpire** attribute and the **histsize** attribute are set, the system retains the number of passwords required to satisfy both conditions, up to the system limit of 50 passwords per user. Null passwords are not retained.

You can edit the **/etc/security/user** file to include any defaults you want to use to administer user passwords. Alternatively, you can change attribute values by using the **chuser** command.

Other commands that can be used with this file are the **mkuser**, **lsuser**, and **rmuser** commands. 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 display the attributes and their values, use the **lsuser** command. To remove a user, use the **rmuser** command.

Extending password restrictions

The rules used by the password program to accept or reject passwords (the password composition restrictions) can be extended by system administrators to provide site-specific restrictions.

Restrictions are extended by adding methods, which are called during a password change. The **pwdchecks** attribute in the **/etc/security/user** file specifies the methods called.

The *AIX 5L Version 5.3 Technical Reference* contains a description of the **pwdrestrict_method**, the subroutine interface to which specified password restriction methods must conform. To correctly extend the password composition restrictions, the system administrator must program this interface when writing a password-restriction method. Use caution in extending the password-composition restrictions. These extensions directly affect the **login** command, the **passwd** command, the **su** command, and other programs. The security of the system could easily be subverted by malicious or defective code.

User authentication

Identification and authentication establish a user's identity.

Each user is required to log in to the system. The user supplies the user name of an account and a password if the account has one (in a secure system, all accounts must either have passwords or be invalidated). If the password is correct, the user is logged in to that account; the user acquires the access rights and privileges of the account. The **/etc/passwd** and **/etc/security/passwd** files maintain user passwords.

By default users are defined in the Files registry. This means that user account and group information is stored in the flat-ASCII files. With the introduction of plug-in load modules, users can be defined in other registries too. For example, when the LDAP plug-in module is used for user administration, then the user definitions are stored in the LDAP repository. In this case there will be no entry for users in the **/etc/security/user** file (there is an exception to this for the user attributes **SYSTEM** and **registry**). When a compound load module (i.e. load modules with an authentication and database part) is used for user administration, the database half determines how AIX user account information is administrated, and the authentication half describes the authentication and password related administration. The authentication half may also describe authentication-specific user account administration attributes by implementing certain load module interfaces (newuser, getentry, putentry etc).

Alternative methods of authentication are integrated into the system by means of the **SYSTEM** attribute that appears in **/etc/security/user** file. The **SYSTEM** attribute allows the system administrator to specify to a fine granularity to which method (or methods) a user must successfully authenticate in order to gain access to the system. For instance, the Distributed Computing Environment (DCE) requires password authentication but validates these passwords in a different manner than the encryption model used in the **/etc/passwd** command and the **/etc/security/passwd** command.

The value of the **SYSTEM** attribute is defined through a grammar. By using this grammar, the system administrators can combine one or more methods to authenticate a particular user to the system. The well known method tokens are `compat`, `DCE`, `files` and `NONE`.

The system default is `compat`. The default `SYSTEM=compat` tells the system to use the local database for authentication and, if no resolution is found, the Network Information Services (NIS) database is tried. The `files` token specifies that only local files are to be used during authentication, whereas `SYSTEM=DCE` results in a `DCE` authentication flow.

The `NONE` token turns off method authentication. To turn off all authentication, the `NONE` token must appear in the `SYSTEM` and `auth1` lines of the user's stanza.

You can specify two or more methods and combine them with the logical constructors `AND` and `OR`. For instance `SYSTEM=DCE` OR `compat` indicates that the user is allowed to login if either `DCE` or local authentication (crypt()) succeeds in this given order.

In a similar fashion a system administrator can use authentication load module names for the **SYSTEM** attribute. For instance when **SYSTEM** attribute is set to `SYSTEM=KRB5files` OR `compat`, the AIX host will first try a Kerberos flow for authentication and if it fails, then it will try standard AIX authentication.

SYSTEM and **registry** attributes are always stored on the local file system in the **/etc/security/user** file. If an AIX user is defined in LDAP and the **SYSTEM** and **registry** attributes are set accordingly, then the user will have an entry in the **/etc/security/user** file.

The **SYSTEM** and **registry** attributes of a user can be changed using the **chuser** command.

Acceptable tokens for the **SYSTEM** attribute can be defined in the **/usr/lib/security/methods.cfg** file.

Note: The root user is always authenticated by means of the local system security file. The **SYSTEM** attribute entry for the root user is specifically set to `SYSTEM=compat` in the**/etc/security/user** file.

For more information on protecting passwords, see the *AIX 5L Version 5.3 System User's Guide: Operating System and Devices*.

Alternative methods of authentication are integrated into the system by means of the **SYSTEM** attribute that appears in **/etc/security/user**. For instance, the Distributed Computing Environment (DCE) requires password authentication but validates these passwords in a manner different from the encryption model used in **etc/passwd** and **/etc/security/passwd**. Users who authenticate by means of DCE can have their stanza in **/etc/security/user** set to `SYSTEM=DCE`.

Other **SYSTEM** attribute values are **compat**, **files**, and **NONE**. The `compat` token is used when name resolution (and subsequent authentication) follows the local database, and if no resolution is found, the Network Information Services (NIS) database is tried. The `files` token specifies that only local files are to be used during authentication. Finally, the `NONE` token turns off method authentication. To turn off all authentication, the `NONE` token must appear in the **SYSTEM** and **auth1** lines of the user's stanza.

Other acceptable tokens for the **SYSTEM** attribute can be defined in **/usr/lib/security/methods.cfg**.

Note: The root user is always authenticated by means of the local system security file. The **SYSTEM** attribute entry for the root user is specifically set to `SYSTEM = "compat"` in **/etc/security/user**.

See the *AIX 5L Version 5.3 System User's Guide: Operating System and Devices* for more information on protecting passwords.

Login user IDs

All audit events recorded for this user are labeled with this ID and can be examined when you generate audit records. See the *AIX 5L Version 5.3 System User's Guide: Operating System and Devices* for more information about login user IDs.

Disk quota system overview

The disk quota system allows system administrators to control the number of files and data blocks that can be allocated to users or groups.

Disk quota system concept

The disk quota system, based on the Berkeley Disk Quota System, provides an effective way to control the use of disk space. The quota system can be defined for individual users or groups, and is maintained for each journaled file system (JFS and JFS2).

The disk quota system establishes limits based on the following parameters that can be changed with the **edquota** command for JFS file systems and the **j2edlimit** command for JFS2 file systems:

- User's or group's soft limits
- User's or group's hard limits
- Quota grace period

The *soft limit* defines the number of 1 KB disk blocks or files under which the user must remain. The *hard limit* defines the maximum amount of disk blocks or files the user can accumulate under the established disk quotas. The *quota grace period* allows the user to exceed the soft limit for a short period of time (the default value is one week). If the user fails to reduce usage below the soft limit during the specified time, the system will interpret the soft limit as the maximum allocation allowed, and no further storage is allocated to the user. The user can reset this condition by removing enough files to reduce usage below the soft limit.

The disk quota system tracks user and group quotas in the **quota.user** and **quota.group** files that reside in the root directories of file systems enabled with quotas. These files are created with the **quotacheck** and **edquota** commands and are readable with the quota commands.

Recovering from over-quota conditions

This section discusses how to recover from over-quota conditions by reducing file system usage.

To reduce file system usage when you have exceeded quota limits, you can use the following methods:

- Kill the current process that caused the file system to reach its limit, remove surplus files to bring the limit below quota, and retry the failed program.
- If you are running an editor such as vi, use the shell escape sequence to check your file space, remove surplus files, and return without losing your edited file. Alternatively, if you are using the C or Korn shells, you can suspend the editor with the Ctrl-Z key sequence, issue the file system commands, and then return with the **fg** (foreground) command.
- Temporarily write the file to a file system where quota limits have not been exceeded, delete surplus files, and then return the file to the correct file system.

Setting up the disk quota system

Typically, only those file systems that contain user home directories and files require disk quotas.

Consider implementing the disk quota system under the following conditions:

- Your system has limited disk space.
- You require more file-system security.
- Your disk-usage levels are large, such as at many universities.

If these conditions do not apply to your environment, you might not want to create disk-usage limits by implementing the disk quota system.

The disk quota system can be used only with the journaled file system.

Note: Do not establish disk quotas for the **/tmp** file system.

To set up the disk quota system, use the following procedure:

1. Log in with root authority.
2. Determine which file systems require quotas.

   Note: Because many editors and system utilities create temporary files in the **/tmp** file system, it must be free of quotas.
3. Use the **chfs** command to include the **userquota** and **groupquota** quota configuration attributes in the **/etc/filesystems** file. The following example uses the **chfs** command to enable user quotas on the **/home** file system:

   ```
   chfs -a "quota = userquota" /home
   ```

   To enable both user and group quotas on the **/home** file system, type:

   ```
   chfs -a "quota = userquota,groupquota" /home
   ```

   The corresponding entry in the **/etc/filesystems** file is displayed as follows:

   ```
   /home:
   dev         = /dev/hd1
   vfs         = jfs
   log         = /dev/hd8
   mount       = true
   check       = true
   quota       = userquota,groupquota
   options     = rw
   ```

4. Optionally, specify alternate disk quota file names. The **quota.user** and **quota.group** file names are the default names located at the root directories of the file systems enabled with quotas. You can specify alternate names or directories for these quota files with the **userquota** and **groupquota** attributes in the **/etc/filesystems** file.

   The following example uses the **chfs** command to establish user and group quotas for the **/home** file system, and names the **myquota.user** and **myquota.group** quota files:

   ```
   chfs -a "userquota = /home/myquota.user" -a "groupquota = /home
           /myquota.group" /home
   ```

   The corresponding entry in the **/etc/filesystems** file is displayed as follows:

   ```
   /home:
   dev        = /dev/hd1
   vfs        = jfs
   log        = /dev/hd8
   mount      = true
   check      = true
   quota      = userquota,groupquota
   userquota  = /home/myquota.user
   groupquota = /home/myquota.group
   options    = rw
   ```
5. If they are not previously mounted, mount the specified file systems.
6. Set the desired quota limits for each user or group. Use the **edquota** command to create each user or group's soft and hard limits for allowable disk space and maximum number of files.

   The following example entry shows quota limits for the *davec* user:

   ```
   Quotas for user davec:
   /home: blocks in use: 30, limits (soft = 100, hard = 150)
           inodes in use: 73, limits (soft = 200, hard = 250)
   ```

   This user has used 30 KB of the maximum 100 KB of disk space. Of the maximum 200 files, *davec* has created 73. This user has buffers of 50 KB of disk space and 50 files that can be allocated to temporary storage.

   When establishing disk quotas for multiple users, use the **-p** flag with the **edquota** command to duplicate a user's quotas for another user.

   To duplicate the quotas established for user *davec* for user *nanc*, type:

   ```
   edquota -p davec nanc
   ```
7. Enable the quota system with the **quotaon** command. The **quotaon** command enables quotas for a specified file system, or for all file systems with quotas (as indicated in the **/etc/filesystems** file) when used with the **-a** flag.
8. Use the **quotacheck** command to check the consistency of the quota files against actual disk usage.

   Note: Do this each time you first enable quotas on a file system and after you reboot the system.

   To enable this check and to turn on quotas during system startup, add the following lines at the end of the **/etc/rc** file:

   ```
   echo " Enabling filesystem quotas "
   /usr/sbin/quotacheck -a
   /usr/sbin/quotaon -a
   ```

Access Control Lists

Typically an ACL consists of series of entries called an Access Control Entry (ACE). Each ACE defines the access rights for a user in relationship to the object.

When an access is attempted, the operating system will use the ACL associated with the object to see whether the user has the rights to do so. These ACLs and the related access checks form the core of the Discretionary Access Control (DAC) mechanism supported by AIX.

The operating system supports several types of system objects that allow user processes to store or communicate information. The most important types of access controlled objects are as follows:

- Files and directories
- Named pipes
- IPC objects such as message queues, shared memory segments, and semaphores

All access permission checks for these objects are made at the system call level when the object is first accessed. Because System V Interprocess Communication (SVIPC) objects are accessed statelessly, checks are made for every access. For objects with file system names, it is necessary to be able to resolve the name of the actual object. Names are resolved either relatively (to the process' working directory) or absolutely (to the process' root directory). All name resolution begins by searching one of these directories.

The discretionary access control mechanism allows for effective access control of information resources and provides for separate protection of the confidentiality and integrity of the information. Owner-controlled access control mechanisms are only as effective as users make them. All users must understand how access permissions are granted and denied, and how these are set.

For example, an ACL associated with a file system object (file or directory) could enforce the access rights for various users in regards to access of the object. It is possible that such an ACL could enforce different levels of access rights, such as read or write, for different users.

Typically, each object will have a defined owner and, in some cases, be associated to a primary group . The owner of a specific object controls its discretionary access attributes. The owner's attributes are set to the creating process's effective user ID.

The following table lists direct access control attributes for the different types of objects:

Owner
For System V Interprocess Communication (SVIPC) objects, the creator or owner can change the object's ownership. SVIPC objects have an associated creator that has all the rights of the owner (including access authorization). The creator cannot be changed, even with root authority.

SVIPC objects are initialized to the effective group ID of the creating process. For file system objects, the direct access control attributes are initialized to either the effective group ID of the creating process or the group ID of the parent directory (this is determined by the group inheritance flag of the parent directory).

Group The owner of an object can change the group. The new group must be either, the effective group ID of the creating process, or the group ID of the parent directory. (As above, SVIPC objects have an associated creating group that cannot be changed, and share the access authorization of the object group.)

Mode The **chmod** command (in numeric mode with octal notations) can set base permissions and attributes. The **chmod** subroutine that is called by the command, disables extended permissions. If you use the numeric mode of the **chmod** command on a file that has an ACL, extended permissions are disabled. The symbolic mode of the **chmod** command does not disable extended permissions. For information about numeric and symbolic mode, see **chmod**.

Many objects in the operating system, such as sockets and file system objects, have ACLs associated for different subjects. Details of ACLs for these object types could vary from one to another.

Traditionally, AIX has supported mode bits for controlling access to the file system objects. It has also supported a unique form of ACL around mode bits. This ACL consisted of base mode bits and also allowed for the definition of multiple ACE entries; each ACE entry defining access rights for a user or group around the mode bits. This classic type of ACL behavior existed prior to AIX 5.3 and will continue to be supported. This ACL type has been named as AIXC ACL type.

Note that support of an ACL on file system objects depends on the underlying physical file system (PFS). The PFS must understand the ACL data and be able to store, retrieve, and enforce the accesses for various users. It is possible that some of the physical file systems do not support any ACLs at all (may just support the base mode bits) as compared to a physical file system that supported multiple types of ACLs. Beginning with AIX 5.3, few of the file systems under AIX have been enhanced to support multiple ACL types. JFS2 and GPFS will have the capability to support NFS version 4 protocol based ACL type too. This ACL has been named NFS4 ACL type on AIX. This ACL type adheres to most of the ACL definition in the NFS version 4 protocol specifications. It also supports more granular access controls as compared to the AIXC ACL type and provides for capabilities such as inheritance.

Multiple Access Control List type framework support

Beginning with version 5.3.0, AIX supports an infrastructure for different Access Control List (ACL) types to exist for different file system objects within the operating system.

This infrastructure allows for uniform methods to manage ACLs irrespective of the ACL type associated with the object. The framework includes the following components:

ACL administration commands
These are commands, such as **aclget**, **aclput**, **acledit**, **aclconvert**, **aclgettypes**. These commands call library interfaces that invoke ACL-type-specific modules.

ACL library interfaces
ACL Library interfaces act as front-ends to the applications that need to access ACLs.

ACL-type-specific dynamically loadable ACL modules
AIX provides a set of ACL-type-specific modules for AIX Classic ACLs (**AIXC**) and NFS4 ACLs (**nfs4**).

Binary compatibility

There are no compatibility issues for applications that run on the existing JFS2 file systems, with or without the existing AIX ACLs.

However, note that applications might find that access to files might fail if they encounter file system objects with much stricter ACLs (such as NFS4) associated. Simple checks to see whether the file exists will require level of read permission in NFS4 ACL.

Access Control List types supported on AIX

AIX currently supports AIXC and NFS4 ACL types.

As mentioned, it also supports an infrastructure for the addition of any other ACL type supported by the underlying physical file system. Note that the JFS2 PFS supports NFS4 ACL natively if the file system instance is created with Extended Attributes Version 2 capability.

AIXC Access Control List

The AIXC Access Control List type represents the behavior of the ACL type supported on AIX releases prior to 5.3.0. AIXC ACLs include base permissions and extended permissions.

The AIXC Access Control List (ACL) type represents the behavior of the ACL type supported on AIX releases prior to 5.3.0. AIXC ACLs include base permissions and extended permissions. The JFS2 file system allows a maximum size of 4 KB for AIXC ACLs.

Setting base permissions for AIXC ACL

Base permissions are the traditional file-access modes assigned to the file owner, file group, and other users. The access modes are: read (r), write (w), and execute/search (x).

In an ACL, base permissions are in the following format, with the *Mode* parameter expressed as rwx (with a hyphen (-) replacing each unspecified permission):

```
base permissions:
   owner(name): Mode
   group(group): Mode
   others: Mode
```

Setting attributes for AIXC ACL

The following attributes can be added to an AIXC ACL:

setuid (SUID)
Set-user-ID mode bit. This attribute sets the effective and saved user IDs of the process to the owner ID of the file at run time.

setgid (SGID)
Set-group-ID mode bit. This attribute sets the effective and saved group IDs of the process to the group ID of the file at run time.

savetext (SVTX)
For directories, indicates that only file owners can link or unlink files in the specified directory.

These attributes are added in the following format:

```
attributes: SUID, SGID, SVTX
```

Setting extended permissions for AIXC Access ACL

Extended permissions allow the owner of a file to more precisely define access to that file. Extended permissions modify the base file permissions (owner, group, others) by permitting, denying, or specifying access modes for specific individuals, groups, or user and group combinations. Permissions are modified through the use of keywords.

The **permit**, **deny**, and **specify** keywords are defined as follows:

permit Grants the user or group the specified access to the file
deny Restricts the user or group from using the specified access to the file
specify Precisely defines the file access for the user or group

If a user is denied a particular access by either a **deny** or a **specify** keyword, no other entry can override that access denial.

The **enabled** keyword must be specified in the ACL for the extended permissions to take effect. The default value is the **disabled** keyword.

In an ACL, extended permissions are in the following format:

```
extended permissions:
  enabled | disabled
    permit   Mode  UserInfo...
    deny     Mode  UserInfo...
    specify  Mode  UserInfo...
```

Use a separate line for each **permit**, **deny**, or **specify** entry. The *Mode* parameter is expressed as **rwx** (with a hyphen (-) replacing each unspecified permission). The *UserInfo* parameter is expressed as `u:UserName`, or `g:GroupName`, or a comma-separated combination of `u:UserName` and `g:GroupName`.

Note: Because a process has only one user ID, if more than one user name is specified in an entry, that entry cannot be used in an access control decision.

Textual representation of AIXC ACL

The following stanza shows the textual representation of an AIXC ACL:

```
Attributes:  { SUID | SGID | SVTX }
Base Permissions:
   owner(name): Mode
   group(group): Mode
   others: Mode
Extended Permissions:
   enabled | disabled
     permit   Mode  UserInfo...
     deny     Mode  UserInfo...
     specify  Mode  UserInfo...
```

Binary format of AIXC ACL

The AIXC ACL binary format is defined in **/usr/include/sys/acl.h** and is implemented in the current AIX release.

AIXC ACL example

The following is an example of an AIXC ACL:

```
attributes: SUID
base permissions:
      owner(frank):  rw-
      group(system): r-x
      others: ---
extended permissions:
      enabled
        permit  rw-  u:dhs
        deny    r--  u:chas, g:system
        specify r--  u:john, g:gateway, g:mail
        permit  rw-  g:account, g:finance
```

The ACL entries are described as follows:

- The first line indicates that the **setuid** bit is turned on.
- The next line, which introduces the base permissions, is optional.
- The next three lines specify the base permissions. The owner and group names in parentheses are for information only. Changing these names does not alter the file owner or file group. Only the **chown** command and the **chgrp** command can change these file attributes.
- The next line, which introduces the extended permissions, is optional.
- The next line indicates that the extended permissions that follow are enabled.
- The last four lines are the extended entries. The first extended entry grants user *dhs* read (r) and write (w) permission on the file.
- The second extended entry denies read (r) access to user *chas* only when he is a member of the *system* group.
- The third extended entry specifies that as long as user *john* is a member of both the *gateway* group and the *mail* group, he has read (r) access. If user *john* is not a member of both groups, this extended permission does not apply.
- The last extended entry grants any user in *both* the *account* group and the *finance* group read (r) and write (w) permission.

Note: More than one extended entry can apply to a process that is requesting access to a controlled object, with restrictive entries taking precedence over permissive modes.

For the complete syntax, see the **acledit** command in the *AIX 5L Version 5.3 Commands Reference*.

NFS4 Access Control List:

AIX also supports the NFS4 Access Control List (ACL) type.

The NFS4 ACL type implements access control as specified in the *Network File System (NFS) version 4 Protocol RFC 3530*. The JFS2 file system allows a maximum size of 64KB for NFS4 ACLs.

Textual representation of NFS4 ACL

A textual NFS V4 ACL is a list of ACEs (Access Control Entries) each ACE per line. An ACE has four elements in the following format.

```
IDENTITY   ACE_TYPE      ACE_MASK     ACE_FLAGS

where:
IDENTITY => Has format of 'IDENTITY_type:(IDENTITY_name or IDENTITY_ID or IDENTITY_who):'
         where:
         IDENTITY_type => One of the following Identity type:
                 u : user
                 g : group
                 s : special who string (IDENTITY_who must be a special who)
                         IDENTITY_name => user/group name
                         IDENTITY_ID   => user/group ID
                         IDENTITY_who  => special who string (e.g. OWNER@, GROUP@, EVERYONE@)
ACE_TYPE => One of the following ACE Type:
                a : allow
                d : deny
                l : alarm
                u : audit
ACE MASK => One or more of the following Mask value Key without separator:
                r : READ_DATA   or LIST_DIRECTORY
                w : WRITE_DATA      or ADD_FILE
                p : APPEND_DATA     or ADD_SUBDIRECTORY
                R : READ_NAMED_ATTRS
                W : WRITE_NAMED_ATTRS
                x : EXECUTE         or SEARCH_DIRECTORY
                D : DELETE_CHILD
                a : READ_ATTRIBUTES
                A : WRITE_ATTRIBUTES
                d : DELETE
                c : READ_ACL
                C : WRITE_ACL
                o : WRITE_OWNER
                s : SYNCHRONIZE
ACE_FLAGS (Optional) => One or more of the following Attribute Key without separater:
                fi : FILE_INHERIT
                di : DIRECTORY_INHERIT
                oi : INHERIT_ONLY
                ni : NO_PROPAGATE_INHERIT
                sf : SUCCESSFUL_ACCESS_ACE_FLAG
                ff : FAILED_ACCESS_ACE_FLAG

Example:
        u:user1(aa@ibm.com):     a    rwp    fidi          * This line is a comment
        *s:(OWNER@):             d    x      dini          * This line is a comment
        g:staff(jj@jj.com):      a    rx
        s:(GROUP@):              a    rwpx   fioi
        u:2:                     d    r     di             * This line shows user bin (uid=2)
        g:7:                     a    ac    fi             * This line shows group security (gid=7)
        s:(EVERYONE@):           a    rca   ni
```

Binary format for NFS4 ACL

The NFS4 ACL binary format is defined in **/usr/include/sys/acl.h** and is implemented in the current AIX release.

NFS4 ACL example

The following example shows an NFS4 ACL applied on a directory (such as, **j2eav2/d0**):

```
s:(OWNER@):       a        rwpRWxDdo       difi        * 1st   ACE
s:(OWNER@):       d        D               difi        * 2nd  ACE
s:(GROUP@):       d        x               ni          * 3rd   ACE
s:(GROUP@):       a        rx              difi        * 4th   ACE
s:(EVERYONE@):    a        c               difi        * 5th   ACE
s:(EVERYONE@):    d        C               difi        * 6th   ACE
u:user1:          a        wp              oi          * 7th   ACE
g:grp1:           d        wp                          *  8th  ACE
u:101:            a        C                           * 9th   ACE
g:100:            d        c                           * 10th  ACE
```

The ACL entries are described as follows:

- The first ACE indicates that the owner has the following privileges on **/j2eav2/d0** and all its offspring created after this ACL is applied:
  - READ_DATA (= LIST_DIRECTORY)
  - WRITE_DATA (=ADD_FILE)
  - APPEND_DATA (= ADD_SUBDIRECTORY)
  - READ_NAMED_ATTR
  - WRITE_NAMED_ATTR
  - EXECUTE (=SEARCH_DIRECTORY)
  - DELETE_CHILD
  - DELETE
  - WRITE_OWNER
- The second ACE indicates the owner is denied the privilege for DELETE_CHILD (deleting the files or subdirectories created under **/j2eav2**), but owner can still delete them because of the first ACE, which allows owner the privilege for DELETE_CHILD.
- The third ACE indicates all members of the group for the object (**/j2eav2/d0**) are denied the privilege for EXECUTE (=SEARCH_DIRECTORY), but the owner is still allowed that privilege by the first ACE. This ACE can not be propagated to all of its offsprings because the NO_PROPAGATE_INHERIT flag is specified. This ACE is applied only to the directory **/j2eav2/d0** and its immediate child files and subdirectories.
- The fourth ACE indicates that every member of the group of the object (**/j2eav2/d0**) is allowed the privilege for READ_DATA (= LIST_DIRECTORY) and EXECUTE (=SEARCH_DIRECTORY) on **/j2eav2/d0** and all its offsprings. However, because of third ACE group members (except the owner) are not allowed the privilege for EXECUTE (=SEARCH_DIRECTORY) on the **/j2eav2/d0** directory and its immediate child files and subdirectories.
- The fifth ACE indicates that everyone is allowed the privilege for READ_ACL on the **/j2eav2/d0** directory and any offspring that are created after this ACL is applied.
- The sixth ACE indicates that everyone is denied the privilege for WRITE_ACL on the **/j2eav2/d0** directory and any offspring. The owner always has the privilege for WRITE_ACL on files and directories with NFS4 ACLs.
- The seventh ACE indicates that user1 has the privilege for WRITE_DATA (=ADD_FILE) and APPEND_DATA (= ADD_SUBDIRECTORY) on all the offspring of the **/j2eav2/d0** directory but not on the **/j2eav2/d0** directory itself.
- The eighth ACE indicates that all the members of grp1 are denied the privilege for WRITE_DATA (=ADD_FILE) and APPEND_DATA (= ADD_SUBDIRECTORY). This ACE does not apply to the owner even it belongs to grp1 because of the first ACE.
- The ninth ACE indicates that the user with **UID 101** has the privilege for WRITE_ACL, but no one, except the owner has the privilege for WRITE_ACL because of the sixth ACE.
- The tenth ACE indicates that all the members of the group with **GID 100** are denied for READ_ACL, but they will have this privilege because of the fifth ACE.

Access Control List Management

This section describes methods of managing ACLs. AIX users can use the Web-based System Manager to view and set ACLs, or they can use the commands.

Applications programmers and other subsystem developers can use the ACL library interfaces and ACL conversion routines described in this section.

ACL administration commands

You can use the following commands to work with ACLs for a file system object:

aclget Writes to standard output the ACL of the file object named *FileObject*, presented in readable format or writes the same to the output file named *outAclFile*.

aclput Sets the ACL of *FileObject* on the file system using the input specified through standard input or *inAclFile*.

acledit
Opens an editor for editing the ACL of the specified *FileObject*.

aclconvert
Converts an ACL from one type to another type. This command fails if the conversion is not supported.

aclgettypes
Gets ACL types supported by a file system path.

ACL library interfaces

ACL Library interfaces act as front-ends to the applications that need to access ACLs. The applications (including the generic ACL administration commands given above) do not directly invoke the undocumented ACL syscalls; instead, they access the generic syscalls and the type-specific loadable modules via the library interfaces. This will shield the customer application programmers from the complexity of using loadable modules, and reduces the backward binary compatibility issues for future AIX releases.

The following library interfaces call syscalls.

aclx_fget and aclx_get
The **aclx_get** and **aclx_fget** functions retrieve the access control information for a file system object, and put it into the memory region specified by **acl**. The size and type information for the **acl** are stored in ***acl_sz** and ***acl_type**.

aclx_fput and aclx_put
The **aclx_put** and **aclx_fput** functions store the access control information specified in **acl** for the input file object. These functions do not do ACL type conversions; for doing ACL type conversion, the caller has to explicitly call the **aclx_convert** function.

aclx_gettypes
The **aclx_gettypes** function gets the list of ACL types supported on the particular file system. A file system type can support more than one ACL type simultaneously. Each file system object is associated with an unique ACL type belonging to the list of ACL types supported by the file system.

aclx_gettypeinfo
The **aclx_gettypeinfo** function gets the characteristics and capabilities of an ACL type on the file system specified by path. Note that the ACL characteristics will normally be of a data structure type, which is specific for each particular ACL type. The data structures used for AIXC and NFS4 ACLs will be described in a separate document.

aclx_print and aclx_printStr
These two functions convert the ACL given in binary format into textual representation. These functions are called by the **aclget** and **acledit** commands.

aclx_scan and aclx_scanStr
These two functions convert the given textual representation of the ACL into binary format.

aclx_convert
Converts an ACL from one type to another. This function is used for implicit conversion by commands, such as **cp**, **mv**, or **tar**.

ACL conversion

ACL conversion allows you to convert one ACL type to another. Support of multiple ACL types is dependent upon what ACL types are support on a specific physical file system. All file systems do not support all ACL types. For example, file system one might support only AIXC ACL types, and file system two might support AIXC and NFS4 ACL types. You can copy AIXC ACLs between the two file systems, but you must use ACL conversion to copy the NFS ACLs from file system two to file system one. ACL conversion preserves the access control information as much as possible.

Note: The conversion process is approximate and could result in loss of access control information. You should consider this when planning your ACL conversions.

ACL conversion in AIX is supported with the following infrastructure:

Library routines
These routines and user level ACL framework enable ACL conversion from one ACL type to another.

aclconvert command
This command converts ACLs.

aclput and acledit commands
These commands are used to modify ACL types.

cp and mv commands
These commands have been enabled to handle multiple ACL types and perform any internal ACL conversion, as necessary.

backup command
This command converts the ACL information to a known type and form (AIXC ACL type), if requested to backup in the legacy format. To retrieve the ACL in its native format, specifiy the **-U** option. See backup for more information.

Each ACL type is unique, and refinement of access control masks varies widely from one ACL type to another. The conversion algorithms are approximate and are not equivalent to manually converting an ACL. In some cases, the conversion will not be exact. For example, NFS4 ACLs cannot truly be converted to AIXC ACLs because NFS4 ACLs provides up to 16 access masks and has inheritance features that are not supported in the AIXC ACL type). You should not use the ACL conversion facilities and interfaces if you are concerned about the loss of access control information.

Note: The ACL conversion algorithms are proprietary in nature and are subject to change.

S bits and Access Control Lists

This section discusses using **setuid** and **setgid** programs and applying S bits to ACLs.

Using setuid and setgid programs

The permission bits mechanism allows effective access control for resources in most situations. But for more precise access control, the operating system provides the **setuid** and **setgid** programs.

AIX defines identity only in terms of uids and gids. ACL types that do not define identity with uids and gids are mapped to the AIX identity model. For example, the NFS4 ACL type defines user identity as strings of the form user@domain, and this string is mapped to numeric UIDs and GIDs.

Most programs execute with the user and group access rights of the user who invoked them. Program owners can associate the access rights of the user who invoked them by making the program a **setuid** or **setgid** program; that is, a program with the setuid or setgid bit set in its permissions field. When that program is executed by a process, the process acquires the access rights of the owner of the program. A **setuid** program executes with the access rights of its owner, while a **setgid** program has the access rights of its group, and both bits can be set according to the permission mechanism.

Although the process is assigned the additional access rights, these rights are controlled by the program bearing the rights. Thus, the **setuid** and **setgid** programs allow for user-programmed access controls in which access rights are granted indirectly. The program acts as a trusted subsystem, guarding the user's access rights.

Although these programs can be used with great effectiveness, there is a security risk if they are not designed carefully. In particular, the program must never return control to the user while it still has the access rights of its owner, because this would allow a user to make unrestricted use of the owner's rights.

Note: For security reasons, the operating system does not support **setuid** or **setgid** program calls within a shell script.

Applying S bits to ACLs

ACLs such as NFS4 do not directly deal with the S bits. NFS4 ACL does not specify how these bits could be accommodated as part of the ACL. AIX has approached the problem such that S bits will be used while performing access checks and will compliment any NFS4 ACL related access checks. AIX's **chmod** command can be used set or reset S bits on file system objects with ACLs such as NFS4.

Administrative access rights

The operating system provides privileged access rights for system administration.

System privilege is based on user and group IDs. Users with effective user or group IDs of 0 are recognized as privileged.

Processes with effective user IDs of 0 are known as root-user processes and can:

- Read or write any object
- Call any system function
- Perform certain subsystem control operations by executing **setuid-root** programs.

You can manage the system using two types of privilege: the **su** command privilege and **setuid-root** program privilege. The **su** command allows all programs you invoke to function as root-user processes. The **su** command is a flexible way to manage the system, but it is not very secure.

Making a program into a **setuid-root** program means the program is a root-user-owned program with the setuid bit set. A **setuid-root** program provides administrative functions that ordinary users can perform without compromising security; the privilege is encapsulated in the program rather than granted directly to the user. It can be difficult to encapsulate all necessary administrative functions in **setuid-root** programs, but it provides more security to system managers.

Access authorization

When a user logs in to an account (using the **login** or **su** commands), the user IDs and group IDs assigned to that account are associated with the user's processes. These IDs determine the access rights of the process.

A process with a user ID of 0 is known as a *root user process*. These processes are generally allowed all access permissions. But if a root user process requests execute permission for a program, access is granted only if execute permission is granted to at least one user.

Access Authorization for AIXC ACLs

The owner of the information resource is responsible for managing access rights. Resources are protected by *permission bits*, which are included in the mode of the object. The permission bits define the access permissions granted to the owner of the object, the group of the object, and for the `others` default class. The operating system supports three different modes of access (read, write, and execute) that can be granted separately.

For files, directories, named pipes, and devices (special files), access is authorized as follows:

- For each access control entry (ACE) in the ACL, the identifier list is compared to the identifiers of the process. If there is a match, the process receives the permissions and restrictions defined for that entry. The logical unions for both permissions and restrictions are computed for each matching entry in the ACL. If the requesting process does not match any of the entries in the ACL, it receives the permissions and restrictions of the default entry.
- If the requested access mode is permitted (included in the union of the permissions) and is not restricted (included in the union of the restrictions), access is granted. Otherwise, access is denied.

The identifier list of an ACL matches a process if all identifiers in the list match the corresponding type of effective identifier for the requesting process. A USER-type identifier matches if it is equal to the effective user ID of the process, and a GROUP-type identifier matches if it is equal to the effective group ID of the process or to one of the supplementary group IDs. For instance, an ACE with an identifier list such as the following:

```
USER:fred, GROUP:philosophers, GROUP:software_programmer
```

would match a process with an effective user ID of *fred* and a group set of:

```
philosophers, philanthropists, software_programmer, doc_design
```

but would not match for a process with an effective user ID of *fred* and a group set of:

```
philosophers, iconoclasts, hardware_developer, graphic_design
```

Note that an ACE with an identifier list of the following would match for both processes:

```
USER:fred, GROUP:philosophers
```

In other words, the identifier list in the ACE functions is a set of conditions that must hold for the specified access to be granted.

All access permission checks for these objects are made at the system call level when the object is first accessed. Because System V Interprocess Communication (SVIPC) objects are accessed statelessly, checks are made for every access. For objects with file system names, it is necessary to be able to resolve the name of the actual object. Names are resolved either relatively (to the process' working directory) or absolutely (to the process' root directory). All name resolution begins by searching one of these directories.

The discretionary access control mechanism allows for effective access control of information resources and provides for separate protection of the confidentiality and integrity of the information. Owner-controlled access control mechanisms are only as effective as users make them. All users must understand how access permissions are granted and denied, and how these are set.

Access Authorization for NFS4 ACLs

Any user who has the privilege for `WRITE_ACL` can control the access rights. The owner of the information resource is always has the privilege for `WRITE_ACL`. For files and directories with NFS4 ACLs, access is authorized as follows:

- The list of ACEs is processed in order and only those ACEs which have a ″who″ (i.e. Identity) that matches the requester are considered for processing. The credentials of the requester is not checked while processing the ACE with special who EVERYONE@.
- Each ACE is processed until all of the bits of requester's access have been allowed. Once a bit is has been allowed, it is no longer considered in the processing of later ACEs.
- If any bit corresponding to the requester's access is denied, access is denied and the remaining ACEs are not processed.
- If all of the bits of requester's access have not been allowed, and there is no ACE left for processing, access is denied.

If the access requested is denied by the ACEs and the requesting user is superuser or root, access is generally allowed. Note that the object owner is always permitted for `READ_ACL`, `WRITE_ACL`, `READ_ATTRIBUTES`, and `WRITE_ATTRIBUTES`. For more information on the algorithm for access authorization, see "NFS4 Access Control List" on page 51.

Access Control List Troubleshooting

This section discusses troubleshooting the Access Control List (ACL).

NFS4 Access Control List on an object failed application

You can use the return code or the trace facility to troubleshoot problems with setting an NFS4 ACL on an object, such as a file or directory. Both methods use command the **aclput** command and the **acledit** command to find the cause of the problem.

Using the Return Code for troubleshooting

To display the return code, use the `echo $?` command after you execute the **aclput** command. The following lists shows the return codes and their explanations:

22 (EINVAL, defined in /usr/include/sys/errno.h)
The following are possible causes for this code:

- Invalid textual format in any field of the 4 fields.
- The size of the input NFS4 ACL is more than 64 KB.
- The ACL is applied on a file that already has at least one ACE with ACE mask set to `w` (`WRITE_DATA`) but not `p` (`APPEND_DATA`) or `p` (`APPEND_DATA`) but not `w` (`WRITE_DATA`).
- The ACL is applied on a directory that already has at least one ACE with ACE mask set to `w` (`WRITE_DATA`) but not `p` (`APPEND_DATA`) or `p` (`APPEND_DATA`) but not `w` (`WRITE_DATA`), and the ACE flag `fi` (`FILE_INHERIT`).
- There is at least one ACE with OWNER@ set as a special **who** (**Identity**) and one or more of the ACE masks `c` (`READ_ACL`), `C` (`WRITE_ACL`), `a` (`READ_ATTRIBUTE`) and `A` (`WRITE_ATTRIBUTE`) are being denied by ACE type `d`.

124 (ENOTSUP, defined in /usr/include/sys/errno.h)
The following are possible causes for this code:

- The special who might not be any one of the three values (`OWNER@`, `GROUP@`, or `EVERYONE@`) in one of the ACEs.

- There is at least one ACE with ACE type `u` (`AUDIT`) or `l` (`ALARM`).

13 (EACCES, defined in /usr/include/sys/errno.h)

The following are possible causes for this code:

- You are not allowed to read the input file containing NFS4 ACEs.
- You are not allowed to search the parent directory of the target object because you do not have `x` (`EXECUTE`) permission on the parent directory of the target object.
- You might not be allowed to write or change the ACL. If the object is already associated with an NFS4 ACL ensure that you are have the privilege for the ACE mask `C` (`WRITE_ACL`).

Using the Trace facility for troubleshooting

You can also generate a trace report to find the cause of the problem. The following scenario shows how to use trace to find the cause of the problem applying an NFS4 ACL. If you have a file, **/j2v2/file1** with the following NFS4 ACL:

```
 s:(EVERYONE@):  a       acC
```

And, the following ACL is contained in the **input_acl_file** input file:

```
s:(EVERYONE@):  a       rwxacC
```

Complete the following steps to troubleshoot with the trace facility:

1. Run the trace, **aclput** and **trcrpt** using the following commands:

   ```
   $ trace -j  478 -o trc.raw
   $->!aclput -i input_acl_file -t NFS4 /j2v2/file1
   $ ->quit
   $ trcrpt trc.raw > trc.rpt
   ```

2. Analyze the trace report. When the ACL is applied on a file or directory, it checks for the access to write or change the ACL, and then applies the ACL. The file contains lines similar to the following:

   ```
   478 xxx  xxx  ACL ENGINE: chk_access entry: type=NFS4 obj_mode=33587200 size=68 ops=16384 uid=100

   478 xxx  xxx  ACL ENGINE: chk_access exit: type=NFS4 rc=0 ops=16384 priv=0 against=0
   478 xxx  xxx  ACL ENGINE: set_acl entry: type=NFS4 ctl_flg=2 obj_mode=33587200 mode=0 size=48

   478 xxx  xxx  ACL ENGINE: validate_acl: type=NFS4 rc=22 ace_cnt=1 acl_len=48 size=12
   478 xxx  xxx  ACL ENGINE: set_acl exit: type=NFS4 rc=22 obj_mode=33587200 size=68 cmd=536878912
   ```

   The second line containing, `chk_access exit`, indicates access is allowed (`rc = 0`) to write the ACL. The fourth line, containing `validate_acl`, and the fifth line, containing `set_acl exit`, indicate that the ACL is not applied successfully (`rc=22` indicates **EINVAL**). The fourth line, containing `validate_acl`, indicates there is problem in the first line of the ACE (`ace_cnt=1`). If you refer to the first ACE, `s:(EVERYONE@): a rwxacC`), there is no **p** as the access mask. The **p** is needed in addition to the **w** when applying the ACL.

Troubleshooting access denies

A filesystem operation (for example, read or write) might fail on an object associated with an NFS4 ACL. Usually, an error message is displayed, but that message might not contain enough information to determine the access problem. You can use the trace facility to find the access problem. For example, if you have a file, **/j2v2/file2** with the following NFS4 ACL:

```
s:(EVERYONE@):  a       rwpx
```

The following command reports a ″Permission denied″ error:

```
ls -l /j2v2/file2
```

Complete the following steps to troubleshoot this problem:

1. Run the trace, **ls -l /j2v2/file2** and trcrpt using the following commands:

```
$ trace -j  478 -o trc.raw
$->!ls -l /j2v2/file2
$ ->quit
$ trcrpt trc.raw > trc.rpt
```

2. Analyze the trace report. The file contains lines similar to the following:

```
478  xxx  xxx   ACL ENGINE: chk_access entry: type=NFS4 obj_mode=33587711 size=68 ops=1024 uid=100
478  xxx  xxx   ACL ENGINE: nfs4_chk_access_self: type=NFS4 aceN=1 aceCnt=1 req=128 deny=0
478  xxx  xxx   ACL ENGINE: nfs4_mask_privcheck: type=NFS4 deny=128 priv=128
478  xxx  xxx   ACL ENGINE: chk_access exit: type=NFS4 rc=13 ops=1024 priv=0 against=0
```

The third line indicates the access is denied for `access mask = 128` (**0x80**) which is only `READ_ATTRIBUTES` (see the **/usr/include/sys/acl.h** file).

Auditing overview

The auditing subsystem enables the system administrator to record security-relevant information, which can be analyzed to detect potential and actual violations of the system security policy.

Auditing subsystem

The auditing subsystem has the following functions:

- “Auditing event detection”
- “Event information collection”
- “Audit trail information processing” on page 60

The system administrator can configure each of these functions.

Auditing event detection

Event detection is distributed throughout the Trusted Computing Base (TCB), both in the kernel (supervisor state code) and the trusted programs (user state code). An auditable event is any security-relevant occurrence in the system. A security-relevant occurrence is any change to the security state of the system, any attempted or actual violation of the system access control or accountability security policies, or both. The programs and kernel modules that detect auditable events are responsible for reporting these events to the system audit logger, that runs as part of the kernel and can be accessed either with a subroutine (for trusted program auditing) or within a kernel procedure call (for supervisor state auditing). The information reported includes the name of the auditable event, the success or failure of the event, and any additional event-specific information that is relevant to security auditing.

Event detection configuration consists of turning event detection on or off, and specifiying which events are to be audited for which users. To activate event detection use the **audit** command to enable or disable the audit subsystem. The **/etc/security/audit/config** file contains the events and users that are processed by the audit subsystem.

Event information collection

Information collection encompasses logging the selected auditable events. This function is performed by the kernel audit logger, which provides both a system call and an intra-kernel procedure call interface that records auditable events.

The audit logger is responsible for constructing the complete audit record, consisting of the audit header, that contains information common to all events (such as the name of the event, the user responsible, the time and return status of the event), and the audit trail, which contains event-specific information. The audit logger appends each successive record to the kernel audit trail, which can be written in either (or both) of two modes:

BIN mode
The trail is written into alternating files, providing for safety and long-term storage.

STREAM mode
The trail is written to a circular buffer that is read synchronously through an audit pseudo-device. STREAM mode offers immediate response.

Information collection can be configured at both the front end (event recording) and at the back end (trail processing). Event recording is selectable on a per-user basis. Each user has a defined set of audit events that are logged in the audit trail when they occur. At the back end, the modes are individually configurable, so that the administrator can employ the back-end processing best suited for a particular environment. In addition, BIN mode auditing can be configured to generate an alert in case the file system space available for the trail is getting too low.

Audit trail information processing

The operating system provides several options for processing the kernel audit trail. The BIN mode trail can be compressed, filtered, or formatted for output, or any reasonable combination of these before archival storage of the audit trail, if any. Compression is done through Huffman encoding. Filtering is done with standard query language (SQL)-like audit record selection (using the **auditselect** command), which provides for both selective viewing and selective retention of the audit trail. Formatting of audit trail records can be used to examine the audit trail, to generate periodic security reports, and to print a paper audit trail.

The STREAM mode audit trail can be monitored in real time, to provide immediate threat-monitoring capability. Configuration of these options is handled by separate programs that can be invoked as daemon processes to filter either BIN or STREAM mode trails, although some of the filter programs are more naturally suited to one mode or the other.

Event selection

Event selection must maintain a balance between insufficient to too much detail.

The set of auditable events on the system defines which occurrences can actually be audited and the granularity of the auditing provided. The auditable events must cover the security-relevant events on the system, as defined previously. The level of detail you use for auditable event definition must maintain a balance between insufficient detail, which makes it difficult for the administrator to understand the selected information, and too much detail, which leads to excessive information collection. The definition of events takes advantage of similarities in detected events. For the purpose of this discussion, a *detected event* is any single instance of an auditable event; for instance, a given event might be detected in various places. The underlying principle is that detected events with similar security properties are selected as the same auditable event. The following list shows a classification of security policy events:

- Subject Events
  - Process creation
  - Process deletion
  - Setting subject security attributes: user IDs, group IDs
  - Process group, control terminal
- Object Events
  - Object creation
  - Object deletion
  - Object open (including processes as objects)
  - Object close (including processes as objects)
  - Setting object security attributes: owner, group, ACL
- Import/Export Events
  - Importing or exporting an object

- Accountability Events
  - Adding a user, changing user attributes in the password database
  - Adding a group, changing group attributes in the group database
  - User login
  - User logoff
  - Changing user authentication information
  - Trusted path terminal configuration
  - Authentication configuration
  - Auditing administration: selecting events and audit trails, switching on or off, defining user auditing classes
- General System Administration Events
  - Use of privilege
  - File system configuration
  - Device definition and configuration
  - System configuration parameter definition
  - Normal system IPL and shutdown
  - RAS configuration
  - Other system configuration
- Security Violations (potential)
  - Access permission refusals
  - Privilege failures
  - Diagnostically detected faults and system errors
  - Attempted alteration of the TCB

Auditing subsystem configuration

The auditing subsystem has a global state variable that indicates whether the auditing subsystem is on. In addition, each process has a local state variable that indicates whether the auditing subsystem should record information about this process.

Both of these variables determine whether events are detected by the Trusted Computing Base (TCB) modules and programs. Turning TCB auditing off for a specific process allows that process to do its own auditing and not to bypass the system accountability policy. Permitting a trusted program to audit itself allows for more efficient and effective collection of information.

Auditing subsystem information collection

Information collection addresses event selection and kernel audit trail modes. It is done by a kernel routine that provides interfaces to log information, used by the TCB components that detect auditable events, and configuration interfaces, used by the auditing subsystem to control the audit logging routine.

Audit logging

Auditable events are logged by the following interfaces: the user state and supervisor state. The user state portion of the TCB uses the **auditlog** or **auditwrite** subroutine, while the supervisor state portion of the TCB uses a set of kernel procedure calls.

For each record, the audit event logger prefixes an audit header to the event-specific information. This header identifies the user and process for which this event is being audited, as well as the time of the event. The code that detects the event supplies the event type and return code or status and optionally, additional event-specific information (the event tail). Event-specific information consists of object names (for example, files that are refused access or tty used in failed login attempts), subroutine parameters, and other modified information.

Events are defined symbolically, rather than numerically. This lessens the chances of name collisions, without using an event registration scheme. Because subroutines are auditable and the extendable kernel definition has no fixed switched virtual circuit (SVC) numbers, it is difficult to record events by number. The number mapping would have to be revised and logged every time that the kernel interface was extended or redefined.

Audit record format

The audit records consist of a common header, followed by audit trails specific to the audit event of the record. The structures for the headers are defined in the **/usr/include/sys/audit.h** file. The format of the information in the audit trails is specific to each base event and is shown in the **/etc/security/audit/events** file.

The information in the audit header is generally collected by the logging routine to ensure its accuracy, while the information in the audit trails is supplied by the code that detects the event. The audit logger has no knowledge of the structure or semantics of the audit trails. For example, when the **login** command detects a failed login, it records the specific event with the terminal on which it occurred and writes the record into the audit trail using the **auditlog** subroutine. The audit logger kernel component records the subject-specific information (user IDs, process IDs, time) in a header and appends this to the other information. The caller supplies only the event name and result fields in the header.

Audit logger configuration

The audit logger is responsible for constructing the complete audit record. You must select the audit events that you want to be logged.

Audit events selection

Audit event selection has the following types:

Per-Process Auditing

To select process events efficiently, the system administrator can define audit classes. An audit class is a subset of the base auditing events in the system. Auditing classes provide for convenient logical groupings of the base auditing events.

For each user on the system, the system administrator defines a set of audit classes that determine the base events that could be recorded for that user. Each process run by the user is tagged with its audit classes.

Per-Object Auditing

The operating system provides for the auditing of object accesses by name; that is, the auditing of specific objects (normally files). By-name object auditing prevents having to cover all object accesses to audit the few pertinent objects. In addition, the auditing mode can be specified, so that only accesses of the specified mode (read/write/execute) are recorded.

Kernel audit trail modes

Kernel logging can be set to BIN or STREAM modes to define where the kernel audit trail is to be written. If the BIN mode is used, the kernel audit logger must be given (before audit startup) at least one file descriptor to which records are to be appended.

BIN mode consists of writing the audit records into alternating files. At auditing startup, the kernel is passed two file descriptors and an advisory maximum bin size. It suspends the calling process and starts writing audit records into the first file descriptor. When the size of the first bin reaches the maximum bin size, and if the second file descriptor is valid, it switches to the second bin and reactivates the calling process. The kernel continues writing into the second bin until it is called again with another valid file descriptor. If at that point the second bin is full, it switches back to the first bin, and the calling process returns immediately. Otherwise, the calling process is suspended, and the kernel continues writing records into the second bin until it is full. Processing continues this way until auditing is turned off. See the following figure for an illustration of audit BIN mode:

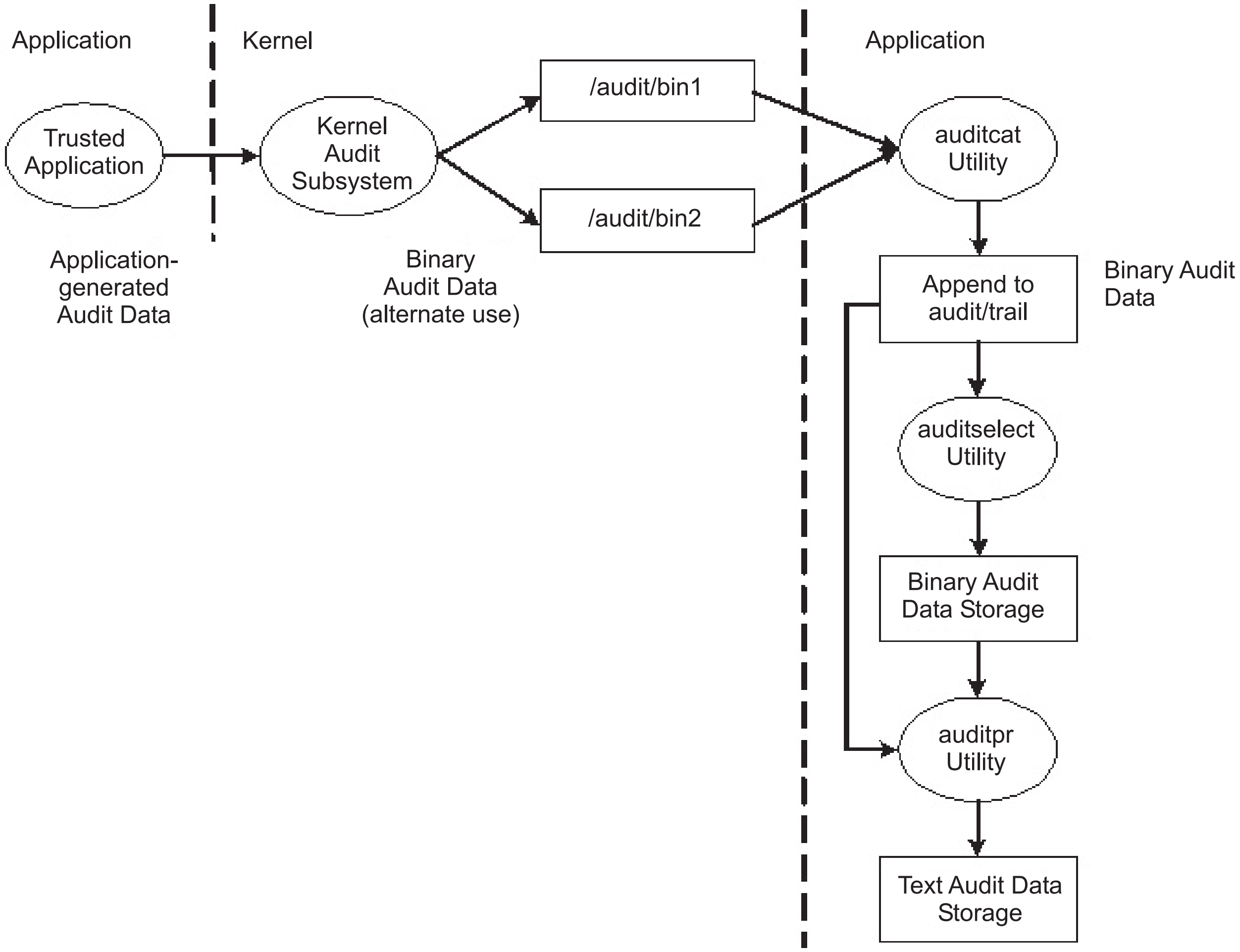


Figure 1. Process of the audit BIN mode.. This illustration shows the process of the audit BIN mode.

The alternating bin mechanism is used to ensure that the audit susbsystem always has something to write to while the audit records are processed. When the audit subsystem switches to the other bin, it empties the first bin content to the **trace** file. When time comes to switch the bin again, the first bin is available. It decouples the storage and analysis of the data from the data generation. Typically, the **auditcat** program is used to read the data from the bin that the kernel is not writing to at the moment. To make sure that the system never runs out of space for the audit trail (the output of the **auditcat** program), the *freespace* parameter can be specified in the **/etc/security/audit/config** file. If the system has less than the amount of 512-byte blocks specified here, it generates a syslog message.

If auditing is enabled, the *binmode* parameter in the `start` stanza in **/etc/security/audit/config** should be set to `panic`. The *freespace* parameter in the `bin` stanza should be configured at minimum to a value that equals 25 percent of the disk space dedicated to the storage of the audit trails. The *bytethreshold* and *binsize* parameters should each be set to 65536 bytes.

In the STREAM mode, the kernel writes records into a circular buffer. When the kernel reaches the end of the buffer, it simply wraps to the beginning. Processes read the information through a pseudo-device called `/dev/audit`. When a process opens this device, a channel is created for that process. Optionally, the events to be read on the channel can be specified as a list of audit classes. See the following figure for an illustration of audit STREAM mode:

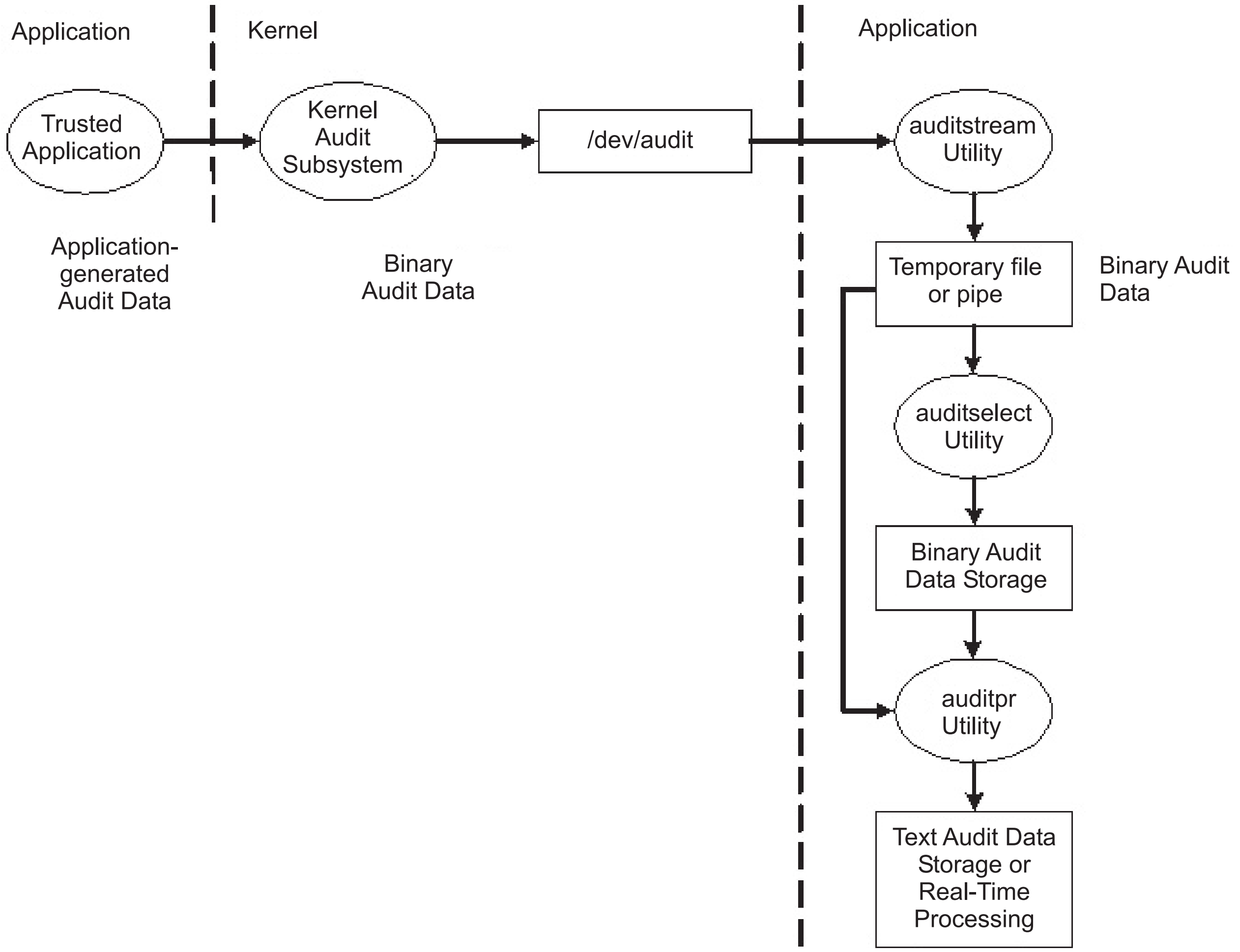


Figure 2. Process of the audit STREAM mode. This illustration shows the process of the audit STREAM mode.

The main purpose of the STREAM mode is to allow for timely reading of the audit trail, which is desirable for real-time threat monitoring. Another use is to create a trail that is written immediately, preventing any possible tampering with the audit trail, as is possible if the trail is stored on some writable media.

Yet another method to use the STREAM mode is to write the audit stream into a program that stores the audit information on a remote system, which allows central near-time processing, while at the same time protecting the audit information from tampering at the originating host.

Audit records processing

The **auditselect**, **auditpr**, and **auditmerge** commands are available to process BIN or STREAM mode audit records. Both utilities operate as filters so that they can be easily used on pipes, which is especially handy for STREAM mode auditing.

auditselect

Can be used to select only specific audit records with SQL-like statements. For example, to select only **exec()** events that were generated by user *afx*, type the following:

```
auditselect -e "login==afx && event==PROC_Execute"
```

auditpr

Used to convert the binary audit records into a human-readable form. The amount of information displayed depends on the flags specified on the command line. To get all the available information, run the **auditpr** command as follows:

```
auditpr -v -hhelrtRpPTc
```

When the **-v** flag is specified, the audit tail which is an event specific string (see the **/etc/security/audit/events** file) is displayed in addition to the standard audit information that the kernel delivers for every event.

auditmerge

Used to merge binary audit trails. This is especially useful if there are audit trails from several systems that need to be combined. The **auditmerge** command takes the names of the trails on the command line and sends the merged binary trail to standard output, so you still need to use the **auditpr** command to make it readable. For example, the **auditmerge** and **auditptr** commands could be run as follows:

```
auditmerge trail.system1 trail.system2 | auditpr -v -hhelrRtpc
```

Using the audit subsystem for a quick security check

To monitor a single suspicious program without setting up the audit subsystem, the **watch** command can be used. It will record either the requested or all events that are generated by the specified program.

For example, to see all **FILE_Open** events when running **vi /etc/hosts**, type the following:

```
watch -eFILE_Open -o /tmp/vi.watch vi /etc/hosts
```

The **/tmp/vi.watch** file displays all **FILE_Open** events for the editor session.

Setting up auditing

The following procedure shows you how to set up an auditing subsystem. For more specific information, refer to the configuration files noted in these steps.

1. Select system activities (events) to audit from the list in the **/etc/security/audit/events** file. If you have added new audit events to applications or kernel extensions, you must edit the file to add the new events.
   - You add an event to this file if you have included code to log that event in an application program (using the **auditwrite** or **auditlog** subroutine) or in a kernel extension (using the **audit_svcstart**, **audit_svcbcopy**, and **audit_svcfinis** kernel services).
   - Ensure that formatting instructions for any new audit events are included in the **/etc/security/audit/events** file. These specifications enable the **auditpr** command to write an audit trail when it formats audit records.
2. Group your selected audit events into sets of similar items called *audit classes*. Define these audit classes in the classes stanza of the **/etc/security/audit/config** file.
3. Assign the audit classes to the individual users and assign audit events to the files (objects) that you want to audit, as follows:
   - To assign audit classes to an individual user, add a line to the users stanza of the **/etc/security/audit/config** file. To assign audit classes to a user, you can use the **chuser** command.
   - To assign audit events to an object (data or executable file), add a stanza for that file to the **/etc/security/audit/objects** file.

- You can also specify default audit classes for new users by editing the **/usr/lib/security/mkuser.default** file. This file holds user attributes that will be used when generating new user IDs. For example, use the general audit class for all new user IDs, as follows:

  ```
  user:
      auditclasses = general
      pgrp = staff
      groups = staff
      shell = /usr/bin/ksh
      home = /home/$USER
  ```

  To get all audit events, specify the ALL class. When doing so on even a moderately busy system, a huge amount of data will be generated. It is typically more practical to limit the number of events that are recorded.

4. In the **/etc/security/audit/config** file, configure the type of data collection that you want using BIN collection, STREAM collection, or both methods. Make sure that audit data does not compete with other data about file space by using a separate file system for audit data. This ensures that there is enough space for the audit data. Configure the type of data collection as follows:
   - To configure BIN collection:
     a. Enable the BIN mode collection by setting binmode = on in the start stanza.
     b. Edit the binmode stanza to configure the bins and trail, and specify the path of the file containing the BIN mode back-end processing commands. The default file for back-end commands is the **/etc/security/audit/bincmds** file.
     c. Make sure that the audit bins are large enough for your needs and set the *freespace* parameter accordingly to get an alert if the file system is filling up.
     d. Include the shell commands that process the audit bins in an audit pipe in the **/etc/security/audit/bincmds** file.
   - To configure STREAM collection:
     a. Enable the STREAM mode collection by setting streammode = on in the start stanza.
     b. Edit the streammode stanza to specify the path to the file containing the streammode processing commands. The default file containing this information is the **/etc/security/audit/streamcmds** file.
     c. Include the shell commands that process the stream records in an audit pipe in the **/etc/security/audit/streamcmds** file.
5. When you have finished making any necessary changes to the configuration files, you are ready to use the **audit start** command option to enable the audit subsystem.
6. Use the **audit query** command option to see which events and objects are audited.
7. Use the **audit shutdown** command option to deactivate the audit subsystem again.

Audit events selection

The purpose of an audit is to detect activities that might compromise the security of your system.

When performed by an unauthorized user, the following activities violate system security and are candidates for an audit:

- Engaging in activities in the Trusted Computing Base
- Authenticating users
- Accessing the system
- Changing the configuration of the system
- Circumventing the auditing system
- Initializing the system
- Installing programs
- Modifying accounts
- Transferring information into or out of the system

The audit system does not have a default set of events to be audited. You must select events or event classes according to your needs.

To audit an activity, you must identify the command or process that initiates the audit event and ensure that the event is listed in the **/etc/security/audit/events** file for your system. Then you must add the event either to an appropriate class in the **/etc/security/audit/config** file, or to an object stanza in the **/etc/security/audit/objects** file. See the **/etc/security/audit/events** file on your system for the list of audit events and trail formatting instructions. For a description of how audit event formats are written and used, see the **auditpr** command.

After you have selected the events to audit, you must combine similar events into audit classes. Audit classes are then assigned to users.

Audit classes selection

You can facilitate the assignment of audit events to users by combining similar events into audit classes. These audit classes are defined in the classes stanza of the **/etc/security/audit/config** file.

Some typical audit classes might be as follows:

general Events that alter the state of the system and change user authentication. Audit attempts to circumvent system access controls.

objects Write access to security configuration files.

kernel Events in the kernel class are generated by the process management functions of the kernel.

An example of a stanza in the **/etc/security/audit/config** file is as follows:

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

Audit data-collection method selection

Your selection of a data-collection method depends on how you intend to use the audit data. If you need long-term storage of a large amount of data, select BIN collection. If you want to process the data as it is collected, select STREAM collection. If you need both long-term storage and immediate processing, select both methods.

Bin collection Allows storage of a large audit trail for a long time. Audit records are written to a file that serves as a temporary bin. After the file is filled, the data is processed by the **auditbin** daemon while the audit subsystem writes to the other bin file, and records are written to an audit trail file for storage.

Stream collection Allows processing of audit data as it is collected. Audit records are written into a circular buffer within the kernel, and are retrieved by reading **/dev/audit**. The audit records can be displayed, printed to provide a paper audit trail, or converted into bin records by the **auditcat** command.

Monitoring file access to critical files in real time

These steps can be used to monitor file access to critical files in real time.

Perform these steps:

1. Set up a list of critical files to be monitored for changes, for example all files in **/etc** and configure them for **FILE_Write** events in the **objects** file:

   ```
   find /etc -type f | awk '{printf("%s:\n\tw = FILE_Write\n\n",$1)}' >> /etc/security/audit/objects
   ```

2. Set up stream auditing to list all file writes. (This example lists all file writes to the console, but in a production environment you might want to have a backend that sends the events into an Intrusion Detection System.) The **/etc/security/audit/streamcmds** file is similar to the following:

```
/usr/sbin/auditstream | /usr/sbin/auditselect -e "event == FILE_Write" |
auditpr  -hhelpPRtTc -v > /dev/console &
```

3. Set up STREAM mode auditing in **/etc/security/audit/config**, add a class for the file write events and configure all users that should be audited with that class:

```
start:
        binmode = off
        streammode = on

stream:
        cmds = /etc/security/audit/streamcmds

classes:
        filemon = FILE_write

users:
        root = filemon
        afx = filemon
        ...
```

4. Now run **audit start**. All **FILE_Write** events are displayed on the console.

Generating a generic audit log

Some examples of generating a generic audit log.

In this example, assume that a system administrator wants to use the audit subsystem to monitor a large multi-user server system. No direct integration into an IDS is performed, all audit records will be inspected manually for irregularities. Only a few essential audit events are recorded, to keep the amount of generated data to a manageable size.

The audit events that are considered for event detection are the following:

FILE_Write	We want to know about file writes to configuration files, so this event will be used with all files in the **/etc** tree.
PROC_SetUserIDs	All changes of user IDs
AUD_Bin_Def	Audit bin configuration
USER_SU	The **su** command
PASSWORD_Change	**passwd** command
AUD_Lost_Rec	Notification in case there where lost records
CRON_JobAdd	new cron jobs
AT_JobAdd	new at jobs
USER_Login	All logins
PORT_Locked	All locks on terminals because of too many invalid attempts

The following is an example of how to generate a generic audit log:

1. Set up a list of critical files to be monitored for changes, such as, all files in **/etc** and configure them for **FILE_Write** events in the **objects** file as follows:

```
find /etc -type f | awk '{printf("%s:\n\tw = FILE_Write\n\n",$1)}' >> /etc/security/audit/objects
```

2. Use the **auditcat** command to set up BIN mode auditing. The **/etc/security/audit/bincmds** file is similar to the following:

```
/usr/sbin/auditcat -p -o $trail $bin
```

3. Edit the **/etc/security/audit/config** file and add a class for the events we have interest. List all existing users and specify the `custom` class for them.

```
start:
        binmode = on
        streammode = off

bin:
        cmds = /etc/security/audit/bincmds
        trail = /audit/trail
        bin1 = /audit/bin1
        bin2 = /audit/bin2
        binsize = 100000
        freespace = 100000

classes:
        custom = FILE_Write,PROC_SetUser,AUD_Bin_Def,AUD_Lost_Rec,USER_SU, \
                 PASSWORD_Change,CRON_JobAdd,AT_JobAdd,USER_Login,PORT_Locked

users:
        root = custom
        afx = custom
        ...
```

4. Add the `custom` audit class to the **/usr/lib/security/mkuser.default** file, so that new IDs will automatically have the correct audit call associated:

   ```
   user:
       auditclasses = custom
       pgrp = staff
       groups = staff
       shell = /usr/bin/ksh
       home = /home/$USER
   ```

5. Create a new file system named **/audit** by using SMIT or the **crfs** command. The file system should be large enough to hold the two bins and a large audit trail.
6. Run the **audit start** command option and examine the **/audit** file. You should see the two bin files and an empty **trail** file initially. After you have used the system for a while, you should have audit records in the **trail** file that can be read with:

   ```
   auditpr  -hhelpPRtTc -v | more
   ```

This example uses only a few events. To see all events, you could specify the classname `ALL` for all users. This action will generate large amounts of data. You might want to add all events related to user changes and privilege changes to your `custom` class.

Light Directory Access Protocol

The Light Directory Access Protocol (LDAP) defines a standard method for accessing and updating information in a directory (a database) either locally or remotely in a client-server model.

The protocol is optimized for reading, browsing, and searching directories, and was originally developed as a lightweight front-end to the X.500 Directory Access Protocol. The LDAP method is used by a cluster of hosts to allow centralized security authentication as well as access to user and group information. This functionality is intended to be used in a clustering environment to keep authentication, user, and group information common across the cluster.

Objects in LDAP are stored in a hierarchical structure known as a Directory Information Tree (DIT). A good directory starts with the structural design of the DIT. The DIT should be designed carefully before implementing LDAP as a means of authentication.

LDAP authentication load module

The LDAP exploitation of the security subsystem is implemented as the LDAP authentication load module. It is conceptually similar to the other load modules such as NIS, DCE, and KRB5. Load modules are defined in the **/usr/lib/security/methods.cfg** file.

The LDAP loadmodule provides user authentication and centralized user and group management functionality through the LDAP protocol. A user defined on a LDAP server can be configured to log in to an LDAP client even if that user is not defined locally.

The AIX LDAP load module is fully integrated within the AIX operating system. After the LDAP authentication load module is enabled to serve user and group information, high-level APIs, commands, and system-management tools work in their usual manner. An **-R** flag is introduced for most high-level commands to work through different load modules. For example, to create an LDAP user named *joe* from a client machine, use the following command:

```
mkuser -R LDAP joe
```

Note: Even though the LDAP infrastructure can support an unlimited number of users in a group, up to 25 000 users have been created in a single group and various operations tested against that group. Some of the historical POSIX interfaces might not return the complete information for the group. Refer to the individual API's documentation for such limitations.

LDAP based authentication

This section describes the limits on the various entities as part of LDAP based authentication on AIX.

Note that LDAP infrastructure itself does not specify any limits on the database contents. However, this section documents the results based on test configurations as to limits. The following limits have been tested with respect to the LDAP based authentication on AIX:

Total number of users: Up to 500 000 users have been created on a single system and simultaneous authentication has been tested for hundreds of users.

Total number of groups: Up to 500 groups have been created on a single system and tested.

Maximum number of users per group: Up to 25 000 users have been created in a single group and various operations tested against that group.

Some of the historical POSIX interfaces might not return the complete information for the group. Refer to the individual API's documentation for such limitations. Also, the above values are based on the testing done. They do not preclude the possibility that one can configure systems with much larger users and groups provided necessary resources exist.

Setting up an LDAP security information server

To set up a system as an LDAP security information server that serves authentication, user, and group information through LDAP, the LDAP server and client packages must be installed.

If the Secure Socket Layer (SSL) is required, the **GSKit** package must be installed. The system administrator must create a key using the **ikeyman** command. For more information about configuring the server to use SSL, see Secure Communication with SSL.

To simplify server configuration, AIX created the **mksecldap** command. The **mksecldap** command can be used to set up an LDAP security information server. It sets up a database named *ldapdb2*, populates the database with the user and group information from the local host, and sets the LDAP server administrator DN (distinguished name) and password. Optionally, it can set up SSL for client/server communication. The **mksecldap** command adds an entry into the **/etc/inittab** file to start the LDAP server at every reboot. The entire LDAP server setup is done through the **mksecldap** command, which updates the **ibmslapd.conf** file (IBM Tivoli® Directory Server Version 5.1 and later) or **slapd.conf** file (SecureWay® Directory Version 3.2 and 4.1) or **slapd32.conf** file (SecureWay Directory Version 3.2).

Unless the `-u NONE` command option for **mksecldap** is used, all users and groups from the local system are exported to the LDAP server during setup. Select one of the following LDAP schemas for this step:

AIX-specific schema

Includes aixAccount and aixAccessGroup object class. This schema offers a full set of attributes for AIX users and groups.

NIS schema (RFC 2307)

Includes posixAccount, shadowAccount, and posixGroup object class and is used by several vendors' directory products. The NIS schema defines only a small subset of attributes that AIX uses.

NIS schema with full AIX support

Includes posixAccount, shadowAccount, and posixGroup object classes plus the aixAusAccount and aixAusGroup object classes. The aixAusAccount and aixAuxGroup object classes provide the attributes which are used by AIX but not defined by the NIS schema. Setting up the LDAP server using NIS schema with full AIX support is recommended unless setting up an AIX-specific schema LDAP server for compatibility with the existing LDAP servers is necessary.

All the user and group information is stored under a common AIX tree (suffix). The default suffix is ″cn=aixdata″. The **mksecldap** command accepts a user-supplied suffix through the **-d** flag. The name for the subtrees to be created for the user, group, ID, and so on is controlled by the **sectoldif.cfg** configuration file. Refer to the **sectoldif.cfg** file for more information.

The created AIX tree is ACL (Access Control List) protected. The default ACL grants administrative privilege only to the entity specified as the administrator with the **-a** command option. Additional privilege can be granted to a proxy identity if the **-x** and **-X** command options are used. Use of these options creates the proxy identity and configure access privilege as defined in the **/etc/security/ldap/proxy.ldif.template** file. Creation of a proxy identity allows LDAP clients to bind to the server without the use of the administrator identity, thereby restricting client administrator privileges on the LDAP server.

The **mksecldap** command works even if an LDAP server has been set up for other purposes; for example, for user ID lookup information. In this case, **mksecldap** adds the AIX tree and populates it with the AIX security information to the existing database. This tree is ACL-protected independently from other trees. In this case, the LDAP server works as usual, in addition to serving as an AIX LDAP Security Server.

Note: Back up the existing database before running the **mksecldap** command to set up the security server to share the same database is recommended.

After the LDAP security information server is successfully set up, the same host can also be set up as a client so that LDAP user and group management can be completed and LDAP users can log in to this server.

If the LDAP security information server setup is not successful, you can undo the setup by running the **mksecldap** command with the **-U** flag. This restores the **ibmslapd.conf** (or **slapd.conf** or **slapd32.conf**) file to its pre-setup state. Run the **mksecldap** command with the **-U** flag after any unsuccessful setup attempt before trying to run the **mksecldap** command again. Otherwise, residual setup information might remain in the configuration file and cause a subsequent setup to fail. As a safety precaution, the undo option does not do anything to the database or to its data, because the database could have existed before the **mksecldap** command was run. Remove any database manually if it was created by the **mksecldap** command. If the **mksecldap** command has added data to a pre-existing database, decide what steps to take to recover from a failed setup attempt.

For more information on setting up an LDAP security information server, see the **mksecldap** command.

Setting up an LDAP client

To set up a client to use LDAP for authentication and user/group information, make sure that each client has the LDAP client package installed. If the SSL is required, the GSKit must be installed, a key must be created, and the LDAP server SSL key certificate must be added to this key.

Similar to LDAP server setup, client setup can be done using the **mksecldap** command. To have this client contact the LDAP security information server, the server name must be supplied during setup. The server's bind DN and password are also needed for client access to the AIX tree on the server. The **mksecldap** command saves the server bind DN, password, server name, AIX tree DN on the server, the SSL key path and password, and other configuration attributes to the **/etc/security/ldap/ldap.cfg** file.

The **mksecldap** command saves the bind password and SSL key password (if configuring SSL) to the **/etc/security/ldap/ldap.cfg** file in encrypted format. The encrypted passwords are system specific, and can only be used by the **secldapclntd** daemon on the system where they are generated. The **secldapclntd** daemon can make use of clear text or encrypted password from the **/etc/security/ldap/ldap.cfg** file.

Multiple servers can be supplied to the **mksecldap** command during client setup. In this case, the client contacts the servers in the supplied order and establishes connection to the first server that the client can successfully bind to. If a connection error occurs between the client and the server, a reconnection request is tried using the same logic. The Security LDAP exploitation model does not support referral. It is important that the replicate servers are kept synchronized.

The client communicates to the LDAP security information server through a client side daemon (**secldapclntd**). If the LDAP load module is enabled on the client, high-level commands are routed to the daemon through the library APIs for users defined in LDAP. The daemon maintains a cache of requested LDAP entries. If a request is not satisfied from the cache, the daemon queries the server, updates the cache, and returns the information back to the caller.

Other fine-tuning options can be supplied to the **mksecldap** command during client setup, such as settings for the number of threads used by the daemon, the cache entry size, and the cache expiration timeout. These options are for experienced users only. For most environments, the default values are sufficient.

In the final steps of the client setup, the **mksecldap** command starts the client-side daemon and adds an entry in the **/etc/inittab** file so the daemon starts at every reboot. You can check whether the setup is successful by checking the **secldapclntd** daemon process through the **ls-secldapclntd** command. Provided that the LDAP security information server is setup and running, this daemon will be running if the setup was successful.

The server must be set up before the client. Client setup depends on the migrated data being on the server. Follow these steps to set up the client:

1. Install **ldap.client** fileset on the AIX 5.3 system.
2. To configure the LDAP client, run the following command:
   ```
   # mksecldap -c -h server1.ibm.com -a cn=admindn -p adminpwd -d cn=basedn
   ```
   Replace the values above as appropriate for your environment.

See the **mksecldap** command description in *AIX 5L Version 5.3 Commands Reference* for more details.

Client enablement for LDAP netgroups:

You can use netgroups as part of NIS-LDAP (the name-resolution method). Perform these steps for client enablement for LDAP netgroups:

1. Install and set up LDAP based user group management as detailed in "Setting up an LDAP client."

If the netgroup setup is not completed, any LDAP-defined user will be listed by the system. For example, if *nguser* is a netgroup user belonging to netgroup *mygroup* already defined in the LDAP server, `lsuser -R LDAP nguser` will list the user.

2. To enable the netgroup function, the module definition for LDAP in the **/usr/lib/security/methods.cfg** file needs to include an options attribute with a netgroup value. Edit the **/usr/lib/security/methods.cfg** file and add the line `options = netgroup` to the LDAP stanza. This marks the LDAP load module as a netgroup-capable load module. For example:

```
LDAP:
      program = /usr/lib/security/LDAP
      program_64 =/usr/lib/security/LDAP64
      options = netgroup
```

   Now the commands `lsuser -R LDAP nguser`, or `lsuser nguser` or `lsuser -R LDAP -a ALL` do not list any users. LDAP is now considered a netgroup-only database from this client and no netgroups have been enabled for access to this client yet.

3. Edit the **/etc/passwd** file, and append a line for the netgroup that should have access to the system. For example if *mygroup* is a netgroup on the LDAP server that contains the desired user, append the following:

```
+@mygroup
```

4. Edit the **/etc/group** file and append a `+:` line to enable NIS lookups for groups:

```
+:
```

   Running the command `lsuser nguser` now returns the user because *nguser* is in the netgroup *mygroup*.

   The `lsuser -R LDAP nguser` command does not find the user, but the command `lsuser -R compat nguser` does because the user is considered a **compat** user now.

5. In order for netgroup users to authenticate to the system, the AIX authentication mechanism must know the method to use. If the default stanza in the **/etc/security/user** file includes `SYSTEM = compat`, then all netgroup users in the netgroup added to the **/etc/passwd** file can authenticate. Another option would be to individually configure users by manually adding stanzas to the **/etc/security/user** file for the desired users. An example stanza for *nguser* is:

```
nguser:
        SYSTEM = compat
        registry = compat
```

   Netgroup users in the allowed netgroups can now authenticate to the system.

   Enabling the netgroup feature also activates the following conditions:

   - Users defined in the **/etc/security/user** file as members of the LDAP registry (having `registry=LDAP` and `SYSTEM="LDAP"`) cannot authenticate as LDAP users. These users are now **nis_ldap** users and require native NIS netgroup membership.
   - The meaning of registry compat is expanded to include modules that use netgroup. For example, if LDAP module is netgroup enabled, compat includes the files, NIS, and LDAP registries. Users retrieved from those modules have a registry value of compat.

LDAP user management

You can manage users and groups on an LDAP security information server from any LDAP client by using high-level commands.

An **-R** flag added to most of the high-level commands can manage users and groups using LDAP as well as other authentication load modules such as DCE, NIS, and KRB5. For more information concerning the use of the **-R** flag, refer to each of the user or group management commands.

To enable a user to authenticate through LDAP, run the **chuser** command to change the user's **SYSTEM** attribute value to LDAP. By setting the **SYSTEM** attribute value according to the defined syntax, a user can be authenticated through more than one load module (for example, compat and LDAP). For more information on setting users' authentication methods, see "User authentication" on page 43 and the **SYSTEM** attribute syntax defined in the **/etc/security/user** file.

A user can become an LDAP user at client setup time by running the **mksecldap** command with the **-u** flag in either of the following forms:

1. Run the command:

   ```
   mksecldap -c -u user1,user2,...
   ```

   where *user1,user2,...* is a list of users. The users in this list can be either locally defined or remote LDAP-defined users. The **SYSTEM** attribute is set to LDAP in each of the above users' stanzas in the **/etc/security/user** file. Such users are only authenticated through LDAP. The users in this list must exist on the LDAP security information server; otherwise, they can not log in from this host. Run the **chuser** command to modify the **SYSTEM** attribute and allow authentication through multiple methods (for example, both local and LDAP).
2. Run

   ```
   mksecldap -c -u ALL
   ```

   This command sets the **SYSTEM** attribute to LDAP in each user's stanza in the **/etc/security/user** file for all locally defined users. All such users only authenticate through LDAP. The locally defined users must exist on the LDAP security information server; otherwise they can not log in from this host. A user that is defined on the LDAP server but not defined locally cannot log in from this host. To allow a remote LDAP-defined user to log in from this host, run the **chuser** command to set the **SYSTEM** attribute to LDAP for that user.

Alternatively, you can enable all LDAP users, whether they are defined locally or not, to authenticate through LDAP on a local host by modifying the ″default″ stanza of the **/etc/security/user** file to use ″LDAP″ as its value. All users that do not have a value defined for their **SYSTEM** attribute must follow what is defined in the default stanza. For example, if the default stanza has `"SYSTEM = "compat""` , changing it to `"SYSTEM = "compat OR LDAP""` allows authentication of these users either through AIX or LDAP. Changing the default stanza to `"SYSTEM = "LDAP""` enables these users to authenticate exclusively through LDAP. Those users who have a **SYSTEM** attribute value defined are not affected by the default stanza.

Setting up SSL on the LDAP server:

In order to set up SSL on the LDAP server, install the **ldap.max_crypto_server** and **GSKit** file sets to enable server encryption support. These file sets can be found on the AIX expansion pack.

Follow these steps to enable SSL support for IBM Directory server authentication.

1. Install the IBM Directory **GSKit** package if it is not installed.
2. Generate the IBM Directory server private key and server certificate using the **gsk7ikm** utility (installed with **GSKit**). The server's certificate might be signed by a commercial Certification Authority (CA), such as VeriSign, or it might be self-signed with the **gsk7ikm** tool. The CA's public certificate (or the self-signed certificate) must also be distributed to the client application's key database file.
3. Store the server's key database file and associated password stash file on the server. The default path for the key database, **/usr/ldap/etc** directory, is a typical location.
4. For initial server setup, run the following command:

   ```
   # mksecldap -s -a cn=admin -p pwd -S rfc2307aix -k /usr/ldap/etc/mykey.kdb -w keypwd
   ```

   Where **mykey.kdb** is the key database, and *keypwd* is the password to the key database. To set up a server that has already been configured and is running:

   ```
   # mksecldap -s -a cn=admin -p pwd -S rfc2307aix -u NONE -k /usr/ldap/etc/mykey.kdb -w keypwd
   ```

Setting up SSL on the LDAP client:

To use SSL on an LDAP client, install the **ldap.max_crypto_client** and **GSKit** filesets off of the AIX expansion pack.

Follow these steps to enable SSL support for LDAP after the server has been enabled for SSL.

1. Run gsk7ikm to generate the key database on each client.
2. Copy the server certificate to each of the clients. If the server SSL uses a self-signed certificate, the certificate must be exported first.
3. On each client system, run gsk7ikm to import the server certificate to the key database.
4. Enable SSL for each client:

   ```
   # mksecldap -c -h servername -a adminDN -p pwd -k /usr/ldap/etc/mykey.kdb -p keypwd
   ```

   Where **/usr/ldap/etc/mykey.kdb** is the full path to the key database and *keypwd* is the password to the key. If the key password is not entered from the command line, a stashed password file from the same directory is used. The stashed file needs to have the same name as the key database with an extension of **.sth** (for example, **mykey.sth**).

LDAP host access control

AIX provides user-level host access (login) control for a system. Administrators can configure LDAP users to log in to an AIX system by setting their **SYSTEM** attribute to LDAP.

The **SYSTEM** attribute is in the **/etc/security/user** file. The **chuser** command can be used to set its value, similar to the following:

```
# chuser -R LDAP SYSTEM=LDAP registry=LDAP foo
```

Note: With this type of control, do not set the default **SYSTEM** attribute to LDAP, which allows all LDAP users to login to the system.

This sets the LDAP attribute to allow user *foo* to log in to this system. It also sets the registry to LDAP, which allows the login process to log *foo*'s login attempts to LDAP, and also allows any user management tasks done on LDAP.

The administrator needs to run such setup on each of the client systems to enable login by certain users.

Starting with AIX 5.2, AIX has implemented a feature to limit a LDAP user only to log in to certain LDAP client systems. This feature allows centralized host access control management. Administrators can specify two host access control lists for a user account: an allow list and a deny list. These two user attributes are stored in the LDAP server with the user account. A user is allowed access to systems or networks that are specified in the allow list, while he is denied access to systems or networks in the deny list. If a system is specified in both the allow list and the deny list, the user is denied access to the system. There are two ways to specify the access lists for a user: with the **mkuser** command when the user is created or with the **chuser** command for a existing user. For backward compatibility, if both the allow list and deny list do not exist for a user, the user is allowed to login to any LDAP client systems by default. Beginning in AIX 5.2, this host access control feature is available.

Examples of setting allow and deny permission lists for users are the following:

```
# mkuser -R LDAP hostsallowedlogin=host1,host2 foo
```

This creates a user *foo*, and user *foo* is only allowed to log in to host1 and host2.

```
# mkuser -R LDAP hostsdeniedlogin=host2 foo
```

This create user *foo*, and user *foo* can log in to any LDAP client systems except host2.

```
# chuser -R LDAP hostsallowedlogin=192.9.200.1 foo
```

This sets user *foo* with permission to log in to the client system at address 192.9.200.1.

```
# chuser -R LDAP hostsallowedlogin=192.9.200/24 hostsdeniedlogin=192.9.200.1 foo
```

This sets user *foo* with permission to log in to any client system within the `192.9.200/24` subnet , except the client system at address `192.9.200.1`.

For more information, see the **chuser** command.

Secure communication with SSL

Depending on the authentication type being used between the LDAP client and server, passwords are sent in either crypted format (unix_auth) or in clear text (ldap_auth). Use Secure Socket Layer (SSL) to protect against security exposure when you send even encrypted passwords over the network, or, in some cases, the Internet. AIX provides packages for SSL that can provide secure communication between directory servers and clients.

For more information, see:

- "Setting up SSL on the LDAP server" on page 74
- "Setting up SSL on the LDAP client" on page 74

Kerberos bind

In addition to a simple bind using a bind DN and a bind password, the **secldapclntd** daemon also supports a bind using Kerberos V credentials.

The keys of the bind principal are stored in a keytab file and need to be made available to the **secldapclntd** daemon in order to use Kerberos bind. With Kerberos bind enabled, the **secldapclntd** daemon does Kerberos authentication to the LDAP server using the principal name and keytab specified in the **/etc/security/ldap/ldap.cfg** client configuration file. Using Kerberos bind makes the **secldapclntd** daemon ignore the bind DN and the bind password specified in **/etc/security/ldap/ldap.cfg** file.

When Kerberos authentication is successful, the **secldapclntd** daemon saves the bind credentials to the **/etc/security/ldap/krb5cc_secldapclntd** directory. The saved credentials are used for a later rebind. If credentials are more than one hour old at the time that the **secldapclntd** daemon tries to rebind to a LDAP server, the **secldapclntd** daemon will reinitialize to renew credentials.

To configure the LDAP client system to use Kerberos bind, you must configure the client using the **mksecldap** command using a bind DN and a bind password. If the configuration is successful, edit the **/etc/security/ldap/ldap.cfg** file with the correct values for Kerberos related attributes. The **secldapclntd** daemon uses the Kerberos bind at restart. After successful configuration, the bind DN and the bind password are not used any more. They can be safely removed or commented out of the **/etc/security/ldap/ldap.cfg** file.

Creating a Kerberos principal:

You need to create at least two principals on the Key Distribution Center (KDC) for use by the IDS server and client in order to support Kerberos bind. The first principal is the LDAP server principal and the second one is the principal used by client systems to bind to the server.

Each of the principal keys need to be placed in a keytab file so that they can be used to start the server process or the client daemon process.

The following example is based on the IBM Network Authentication Service. If you install Kerberos software from other sources, the actual commands may be different than what is shown here.

- Start the kadmin tool on the KDC server as the root user.

  ```
  #/usr/krb5/sbin/kadmin.local
  kadmin.local:
  ```

- Create the ldap/*serverhostname* principal for the LDAP server. The *serverhostname* is the fully qualified DNS host that will run the LDAP server.

```
kadmin.local: addprinc ldap/plankton.austin.ibm.com
WARNING: no policy specified for "ldap/plankton.austin.ibm.com@ud3a.austin.ibm.com":
Re-enter password for principal "ldap/plankton.austin.ibm.com@ud3a.austin.ibm.com":
Principal "ldap/plankton.austin.ibm.com@ud3a.austin.ibm.com" created.
kadmin.local:
```

- Create a keytab for the created server principal. This key will be used by the LDAP server during server startup. To create a keytab called **slapd_krb5.keytab**:

```
kadmin.local: ktadd -k /etc/security/slapd_krb5.keytab ldap/plankton.austin.ibm.com
Entry for principal ldap/plankton.austin.ibm.com with kvno 2,
encryption type Triple DES cbc mode with HMAC/sha1 added to keytab
WRFILE:/etc/security/slapd_krb5.keytab.
Entry for principal ldap/plankton.austin.ibm.com with kvno 2,
encryption type ArcFour with HMAC/md5 added to keytab WRFILE:/etc/security/slapd_krb5.keytab.
Entry for principal ldap/plankton.austin.ibm.com with kvno 2,
encryption type AES-256 CTS mode with 96-bit SHA-1 HMAC added to keytab
WRFILE:/etc/security/slapd_krb5.keytab.
Entry for principal ldap/plankton.austin.ibm.com with kvno 2,
encryption type DES cbc mode with RSA-MD5 added to keytab WRFILE:/etc/security/slapd_krb5.keytab.
kadmin.local:
```

- Create a principal named ldapadmin for the IDS administrator.

```
kadmin.local: addprinc ldapadmin
WARNING: no policy specified for ldapadmin@ud3a.austin.ibm.com; defaulting to no policy.
Note that policy may be overridden by ACL restrictions.
Enter password for principal "ldapadmin@ud3a.austin.ibm.com":
Re-enter password for principal "ldapadmin@ud3a.austin.ibm.com":
Principal "ldapadmin@ud3a.austin.ibm.com" created.
kadmin.local:
```

- Create a keytab for the bind principal **kdapadmin.keytab**. This key can be used by the **secldapclntd** client daemon.

```
kadmin.local: ktadd -k /etc/security/ldapadmin.keytab ldapadmin
Entry for principal ldapadmin with kvno 2, encryption type
Triple DES cbc mode with HMCA/sha1 added to keytab WRFILE:/etc/security/ldapadmin.keytab.
Entry for principal ldapadmin with kvno 2, encryption type
ArcFour with HMAC/md5 added to keytab WRFILE:/etc/security/ldapadmin.keytab.
Entry for principal ldapadmin with kvno 2, encryption type
AES-256 CTS mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/security/ldapadmin.keytab.
Entry for principal ldapadmin with kvno 2, encryption type
DES cbc mode with RSA-MD5 added to keytab WRFILE:/etc/security/ldapadmin.keytab.
kadmin.local
```

- Create a principal named ldapproxy for clients to bind to the LDAP server.

```
kadmin.local: addprinc ldapproxy
WARNING: no policy specified for ldapproxy @ud3a.austin.ibm.com; defaulting to no policy.
Note that policy may be overridden by ACL restriction
Enter password for principal "ldapproxy@ud3a.austin.ibm.com":
Re-enter password for principal "ldapproxy@ud3a.austin.ibm.com":
Principal "ldapproxy@ud3a.austin.ibm.com" created.
kadmin.local:
```

- Create a keytab called *ldapproxy.keytab* for the bind principal **ldapproxy**. This key can be used by the **secldapclntd** client daemon.

```
kadmin.local: ktadd -k /etc/security/ldapproxy.keytab ldapproxy
Entry for principal ldapproxy with kvno 2, encryption type
Triple DES cbc mode with HMAC/sh1 added to keytab WRFILE:/etc/security/ldapproxy.keytab.
Entry for principal ldapproxy with kvno 2, encryption type
ArcFour with HMAC/md5 added to keytab WRFILE:/etc/security/ldapproxy.keytab
Entry for principal ldapproxy with kvno 2, encryption type
AES-256 CTS mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/security/ldapproxy.keytab
Entry for principal ldapproxy with kvno 2,
encryption type DES cbc mode with RSA-MD5 added to keytab WRFILE:/etc/security/ldapproxy.keytab.
kadmin.local:
```

Enabling the IDS server Kerberos bind:

Procedure for enabling the IDS server for Kerberos bind.

The following example shows how to configure an IDS server for Kerberos bind.

This example was tested using IDS v5.1:

1. Install the **krb5.client** fileset.
2. Make sure the **/etc/krb5/krb5.conf** file exists and is configured properly. If you need to configure it, you can run the **/usr/sbin/config.krb5** command.
   ```
   # config.krb5 -r ud3a.austin.ibm.com -d austin.ibm.com -c KDC -s alyssa.austin.ibm.com
   Initializing configuration...
   Creating /etc/krb5/krb5_cfg_type...
   Creating /etc/krb5/krb5.conf...
   The command completed successfully.
   # cat /etc/krb5/krb5.conf
   [libdefaults]
        default_realm = ud3a.austin.ibm.com
        default_keytab_name = FILE:/etc/krb5/krb5.keytab
        default_tkt_enctypes = des3-cbc-sha1 arcfour-hmac aes256-cts des-cbc-md5 des-cbc-crc
        defaut_tgs_enctypes = des3-cbc-shal1 arcfour-hmac aes256-cts des-cbc-md5 des-cbc-crc
   [realms]
     ud3a.austin.ibm.com = {
        kdc = alyssa.austin.ibm.com:88
        admin_server = alyssa.austin.ibm.com:749
        default_domain = austin.ibm.com
     }

   [domain_realm]
        .austin.ibm.com = ud3a.austin.ibm.com
        alyssa.austin.ibm.com = ud3a.austin.ibm.com

   [logging]
        kdc = FILE:/var/krb5/log/krb5
        admin_server = FILE:/var/krb5/log/kadmin.log
        default = FILE:/var/krb5/log/krb5lib.log
   ```
3. Get the keytab file of the ldap:/*serverhostname* principal, and place it in the **/usr/ldap/etc** directory. For example: **/usr/ldap/etc/slapd_krb5.keytab**.
4. Set the permission to allow the server process to access the file.
   ```
   # chown ldap:ldap/usr/ldap/etc/slapd_krb5.keytab
   #
   ```
5. To enable the IDS server for Kerberos bind, edit the **/etc/ibmslapd.conf** file and append the following entry:
   ```
   dn: cn=Kerberos, cn-Configuration
   cn: Kerberos
   ibm-slapdKrbAdminDN: ldapadmin
   ibm-slapdKrbEnable: true
   ibm-slapdKrbIdentityMap: true
   ibm-slapdKrbKeyTab: /usr/ldap/etc/slapd_krb5.keytab
   ibm-slapdKrbRealm: ud3a.austin.ibm.com
   objectclass: ibm-slapdKerberos
   objectclass: ibm-slapdconfigEntry
   objectclass: top
   ```
6. Map the ldapproxy principal to a bind DN named cn-proxyuser,cn=aixdata.
   1. If the bind DN entry exists in the IDS server, create a file named **ldapproxy.ldif** with the following content:
      ```
      dn: cn=proxyuser,cn=aixdata
      changetype: modify
      add: objectclass
      objectclass: ibm-securityidentities
      ```

```
-
add:altsecurityidentities
alsecurityidentities: Kerberos:ldapproxy@ud3a.austin.ibm.com
```

OR

b. If the bind DN entry is not yet added to the server, create a file named **proxyuser.ldif** with the following content:

Note: You will need to replace *proxyuserpwd* with your password.

```
dn: cn=proxyuser,cn=mytest
cn: proxyuser
sn: proxyuser
userpassword: proxyuserpwd
objectclass: person
objectclass: top
objectclass: ibm-securityidentities
altsecurityidentities: Kerberos:ldapproxy@ud3a.austin.ibm.com
```

Add the bind DN entry that is created to the IDS server using the **ldapmodify** command.

```
# ldapmodify -D cn-admin -w adminPwd -f /tmp/proxyuser.ldif modifying entry cn=proxyuser,cn=mytest
#
```

7. Restart the IDS server.

Enabling the AIX LDAP client Kerberos bind:

This section describes how to configure an AIX LDAP client system to use Kerberos in its initial bind to an LDAP server.

The IDS server must be configured in this manner for the server host to be a client to itself.

This example was tested using IDS v 5.1:

1. Install the **krb5.client** fileset.
2. Make sure the **/etc/krb.conf** file exists and is configured properly. If it is not properly configured, you can run the **/usr/sbin/config.krb5** command to configure it.
3. Get the keytab file of the bind principal, and place it in the **/etc/security/ldap** directory.
4. Set the permission to 600.
5. Configure the client using the **mksecldap** command using the bind DN and the bind password. Make sure that AIX commands work on LDAP users.
6. Edit the **/etc/security/ldap/ldap.cfg** file to set the Kerberos related attributes. In the following example, the bind principal is **ldapproxy** and the keytab file is **ldapproxy.keytab**. If you want IDS server administrator privileges, replace the *ldapproxy* with *ldapadmin* and replace the *ldapproxy.keytab* with *ldapadmin.keytab*.

   ```
   useKRB5:yes
   krbprincipal:ldapproxy
   krbkeypath:/etc/security/ldap/ldapproxy.keytab
   krbcmddir:/usr/krb5/bin/
   ```

   Now the bind DN and bind password can be removed or commented out of the **ldap.cfg** file because the **secldapclntd** daemon now uses Kerberos bind.
7. Restart the **secldapclntd** daemon.
8. The **/etc/security/ldap/ldap.cfg** file can now be propagated to other client systems.

LDAP security information server auditing

SecureWay Directory version 3.2 (and later) provides a default server audit logging function. Once enabled, this default audit plugin logs LDAP server activities to a log file. See the LDAP documentation in *Packaging Guide for LPP Installation* for more information on this default audit plugin.

An LDAP security information server auditing function has been implemented in AIX 5.1 and later, called the *LDAP security audit plugin*. It is independent of the SecureWay Directory default auditing service, so that either one or both of these auditing subsystems can be enabled. The AIX audit plugin records only those events that update or query the AIX security information on an LDAP server. It works within the framework of AIX system auditing.

To accommodate LDAP, the following audit events are contained in the **/etc/security/audit/event** file:

- `LDAP_Bind`
- `LDAP_Unbind`
- `LDAP_Add`
- `LDAP_Delete`
- `LDAP_Modify`
- `LDAP_Modifydn`
- `LDAP_Search`

An `ldapserver` audit class definition is also created in the **/etc/security/audit/config** file that contains all of the above events.

To audit the LDAP security information server, add the following line to each user's stanza in the **/etc/security/audit/config** file:

```
ldap = ldapserver
```

Because the LDAP security information server audit plug-in is implemented within the frame of the AIX system auditing, it is part of the AIX system auditing subsystem. Enable or disable the LDAP security information server audit using system audit commands, such as **audit start** or **audit shutdown**. All audit records are added to the system audit trails, which can be reviewed with the **auditpr** command. For more information, see "Auditing overview" on page 59.

LDAP commands

Listing and description of the LDAP commands.

lsldap command

The **lsldap** command can be used to display naming service entities from the configured LDAP server. These entities are aliases, automount, bootparams, ethers, groups, hosts, netgroups, networks, passwd, protocols, rpc and services.

mksecldap command

The **mksecldap** command can be used to set up IBM SecureWay Directory servers and clients for security authentication and data management. This command must be run on the server and all clients.

secldapclntd daemon

The **secldapclntd** daemon accepts requests from the LDAP load module, forwards the request to the LDAP Security Information Server, and passes the result from the server back to the LDAP load module.

For more information on the LDAP attribute mapping file format, see **LDAP attribute mapping file format** in the *AIX 5L Version 5.3 Files Reference*.

Related information

The **mksecldap**, **start-secldapclntd**, **stop-secldapclntd**, **restart-secldapclntd**, **ls-secldapclntd**, **sectoldif**, and **flush-secldapclntd** commands.

The **secldapclntd** daemon.

The **/etc/security/ldap/ldap.cfg** file.

The LDAP attribute mapping file format.

Migration to LDAP from NIS, including the netgroup setting can be found in the Network Information Services (NIS and NIS+) Guide: Appendix B. Migrating from NIS and NIS+ to RFC 2307-compliant LDAP services.

LDAP management commands:

Listing and description of the LDAP management commands.

start-secldapclntd command

The **start-secldapclntd** command starts the **secldapclntd** daemon if it is not running.

stop-secldapclntd command

The **stop-secldapclntd** command terminates the running **secldapclntd** daemon process.

restart-secldapclntd command

The **restart-secldapclntd** script stops the **secldapclntd** daemon if it is running, and then restarts it. If the **secldapclntd** daemon is not running, it simply starts it.

ls-secldapclntd command

The **ls-secldapclntd** command lists the **secldapclntd** daemon status.

flush-secldapclntd command

The **flush-secldapclntd** command clears the cache for the **secldapclntd** daemon process.

sectoldif command

The **sectoldif** command reads users and groups defined locally, and prints the result to standard output in **ldif** format.

ldap.cfg file format:

The **/etc/security/ldap/ldap.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/ldap.cfg** file is updated by the **mksecldap** command at client setup.

For more information on the **/etc/security/ldap/ldap.cfg** file, see **/etc/security/ldap/ldap.cfg** in the *AIX 5L Version 5.3 Files Reference*.

Mapping file format for LDAP attributes:

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. An entry has four space-separated fields:

```
AIX_Attribute_Name AIX_Attribute_Type LDAP_Attribute_Name LDAP_Value_Type
```

AIX_Attribute_Name Specifies the AIX attribute name.

AIX_Attribute_Type Specifies the AIX attribute type. Values are SEC_CHAR, SEC_INT, SEC_LIST, and SEC_BOOL.

LDAP_Attribute_Name Specifies the LDAP attribute name.

LDAP_Value_Type Specifies the LDAP value type. Values are **s** for single value and **m** for multi-value.

Public Key Cryptography Standards #11

The Public Key Cryptography Standards #11 (PKCS #11) subsystem provides applications with a method for accessing hardware devices (tokens) regardless of the type of device.

The content in this section conforms to Version 2.01 of the PKCS #11 standard.

The PKCS #11 subsystem has been implemented using the following components:

- A slot manager daemon (**pkcsslotd**), which provides the subsystem with information regarding the state of available hardware devices. This daemon is started automatically during installation and when the system is rebooted.
- An API shared object (**/usr/lib/pkcs11/pkcs11_API.so**) is provided as a generic interface to the adapters for which PKCS #11 support has been implemented.
- An adapter-specific library, which provides the PKCS #11 support for the adapter. This tiered design allows the user to use new PKCS #11 devices when they come available without recompiling existing applications.

IBM 4758 Model 2 Cryptographic Coprocessor

The IBM 4758 Model 2 Cryptographic Coprocessor provides a secure computing environment.

Before attempting to configure the PKCS #11 subsystem, verify that the adapter has been properly configured with a supported microcode.

IBM 4960 Cryptographic Accelerator

The IBM 4960 Cryptographic Accelerator provides a means of offloading cryptographic transactions. Before attempting to configure the PKCS #11 subsystem, verify that the adapter has been properly configured.

Verifying the IBM 4758 Model 2 Cryptographic Coprocessor for use with the Public Key Cryptography Standards #11 subsystem

The PKCS #11 subsystem is designed to automatically detect adapters capable of supporting PKCS #11 calls during installation and at reboot. For this reason, any IBM 4758 Model 2 Cryptographic Coprocessor that is not properly configured will not be accessible from the PKCS #11 interface and calls sent to the adapter will fail.

To verify that your adapter is set up correctly, complete the following:

1. Ensure that the software for the adapter is properly installed by typing the following command:

   ```
   lsdev -Cc adapter | grep crypt
   ```

   If the IBM 4758 Model 2 Cryptographic Coprocessor is not included in the resulting list, check that the card is seated properly and that the supporting software is correctly installed.
2. Determine that the proper firmware has been loaded onto the card by typing the following:

```
csufclu /tmp/l ST device_number_minor
```

Verify that the Segment 3 Image has the PKCS #11 application loaded. If it is not loaded refer to the adapter specific documentation to obtain the latest microcode and installation instructions.

Note: If this utility is not available, then the supporting software has not been installed.

Verifying the IBM 4760 Model 2 Cryptographic Accelerator for use with the Public Key Cryptography Standards #11 subsystem

The PKCS #11 subsystem is designed to automatically detect adapters capable of supporting PKCS #11 calls during installation and at reboot. For this reason, any IBM 4760 Cryptographic Accelerator that is not properly configured will not be accessible from the PKCS #11 interface and calls sent to the adapter will fail.

To ensure that the software for the adapter is properly installed, type the following command:

```
lsdev -Cc adapter | grep ica
```

If the IBM 4760 Cryptographic Accelerator is not included in the resulting list, check that the card is seated properly and that the supporting device driver is correctly installed.

Public Key Cryptography Standards #11 subsystem configuration

The PKCS #11 subsystem automatically detects devices supporting PKCS #11. However, in order for some applications to use these devices, some initial set up is necessary.

These tasks can be performed through the API (by writing a PKCS #11 application) or by using the SMIT interface. The PKCS #11 SMIT options are accessed either through **Manage the PKCS11 subsystem** from the main SMIT menu, or by using the **smit pkcs11** fast path.

Initializing the token

Each adapter or PKCS #11 token must be initialized before it can be used successfully.

This initialization procedure involves setting a unique label to the token. This label allows applications to uniquely identify the token. Therefore, the labels should not be repeated. However; the API does not verify that labels are not re-used. This initialization can be done through a PKCS #11 application or by the system administrator using SMIT. If your token has a Security Officer PIN, the default value is set to 87654321. To ensure the security of the PKCS #11 subsystem, this value should be changed after initialization.

To initialize the token:

1. Enter the token management screen by typing `smit pkcs11`.
2. Select **Initialize a Token**.
3. Select a PKCS #11 adapter from the list of supported adapters.
4. Confirm your selection by pressing Enter.

   Note: This will erase all information on the token.
5. Enter the Security Officer PIN (SO PIN) and a unique token label.

If the correct PIN is entered, the adapter will be initialized or reinitialized after the command has finished running.

Setting the security officer PIN

Follow these steps to change an SO PIN from its default value.

To change the PIN from its default value:

1. Type `smit pkcs11`.
2. Select **Set the Security Officer PIN**.
3. Select the initialized adapter for which you want to set the PIN.
4. Enter the current PIN and a new PIN.
5. Verify the new PIN.

Initializing the user PIN

After the token has been initialized, it might be necessary to set the user PIN to allow applications to access token objects.

Refer to your device specific documentation to determine if the device requires a user to log in before accessing objects.

To initialize the user PIN:

1. Enter the token management screen typing `smit pkcs11`.
2. Select **Initialize the User PIN**.
3. Select a PKCS #11 adapter from the list of supported adapters.
4. Enter the SO PIN and the User PIN.
5. Verify the User PIN.
6. Upon verification, the User PIN must be changed.

Resetting the user PIN:

To reset the user PIN, you can either reinitialize the PIN using the SO PIN or set the user PIN by using the existing user PIN.

To reset the PIN:

1. Enter the token management screen by typing `smit pkcs11`.
2. Select **Set the User PIN**.
3. Select the initialized adapter for which you want to set the user PIN.
4. Enter the current user PIN and a new PIN.
5. Verify the new user PIN.

Public Key Cryptography Standards #11 usage

For an application to use the PKCS #11 subsystem, the subsystem's slot manager daemon must be running and the application must load in the API's shared object.

The slot manager is normally started at boot time by **inittab** calling the **/etc/rc.pkcs11** script. This script verifies the adapters in the system before starting the slot manager daemon. As a result, the slot manager daemon is not available before the user logs on to the system. After the daemon starts, the subsystem incorporates any changes to the number and types of supported adapters without intervention from the systems administrator.

The API can be loaded either by linking in the object at runtime or by using deferred symbol resolution. For example, an application can get the PKCS #11 function list in the following manner:

```
d CK_RV (*pf_init)();
void *d;
CK_FUNCTION_LIST *functs;

d = dlopen(e, RTLD_NOW);
if ( d == NULL ) {
   return FALSE;
}
```

```
pfoo = (CK_RV (*)())dlsym(d, "C_GetFunctionList");
if (pfoo == NULL) {
   return FALSE;
}

rc = pf_init(&functs);
```

X.509 Certificate Authentication Service and Public Key Infrastructure

Certificate Authentication Service provides the AIX operating system with the ability to authenticate users using X.509 Public Key Infrastructure (PKI) certificates and to associate certificates with processes as proof of a user's identity. It provides this capability through the Loadable Authentication Module Framework (LAMF), the same extensible AIX mechanism used to provide DCE, Kerberos, and other authentication mechanisms.

Overview of Certificate Authentication Service

Every user account participating in PKI authentication has a unique PKI certificate. The certificate in conjunction with a password is used to authenticate the user during login.

PKI certificates are based on public key/private key technology. This technology uses two asymmetric keys to encrypt and decrypt data. Data encrypted using one key can only be decrypted using the other key. A user keeps one key private, called the private key, storing it in a private keystore while publishing the other key, called the public key, in the form of a certificate. Certificates are commonly maintained on a Lightweight Directory Access Protocol (LDAP) server, either within an organization for intra-company usage or on the Internet for world-wide usage.

For a user named John to send a user named Kathy data that only she can decrypt, John would obtain the public key from Kathy's published certificate, encrypt the data using Kathy's public key, and send the data to her. Kathy would decrypt the data from John using her private key located in her private keystore.

This technology is also used for digital signatures. If Kathy wants to send data to John that is digitally signed by her, Kathy would use her private key to digitally sign the data and send the data and digital signature to John. John would obtain the public key from Kathy's published certificate and use the public key to verify the digital signature before using the data.

In both cases, Kathy's private key is maintained in a private keystore. The many types of private keystores include smart cards and files, but all keystore types protect private keys through the use of passwords or Personal Identification Numbers (PINs). They typically provide storage for multiple private keys along with certificates and other PKI objects. Users typically have their own keystores.

Certificate authentication service uses digital-signature technology to authenticate a user during login. Certificate authentication service locates the user's certificate and keystore based off the user's account name, obtains the certificate's matching private key from the user's keystore using the user's password, signs a data item with the user's private key, and checks the signature using the user's public key from the certificate. After the user authenticates, the system stores the user's certificate in protected memory, associating the certificate with every process created by the user. This in-memory association enables quick access to the user's certificate for any process owned by the user, as well as by the operating system's kernel.

Certificates

Understanding certificate authentication service requires a basic understanding of certificates, certificate formats, and certificate lifecycle management.

Certificates are standardized objects that follow the X.509 standard, of which version 3 (X.509v3) is the latest version. Certificates are created, signed, and issued by a Certificate Authority (CA) which is most commonly a software application that accepts and processes certificate requests. Certificates are comprised of several certificate attributes. Some of the attributes are required, but many are optional. Certificate attributes commonly used and discussed in this document are:

- Certificate Version - The X.509 version number (that is, 1, 2, or 3).
- Serial Number - A certificate serial number that uniquely distinguishes the certificate from all other certificates issued by the same CA.
- Issuer Name - A name specifying the certificate's issuing CA.
- Validity Period - The activation and expiration date of the certificate.
- Public Key - The public key.
- Subject Distinguished Name - A name specifying the certificate's owner.
- Subject Alternate Name Email - The owner's email address.
- Subject Alternate Name URI - The owner's Web site URI/URL.

Each certificate has a unique version number that indicates with which version of the X.509 standard it conforms. Each certificate has a serial number which uniquely distinguishes it from all other certificates issued by the same CA. The serial number is unique only to the issuing CA. The certificate's issuer name identifies the issuing CA.

Certificates are valid only between two specified dates: the ″Not Before″ date and the ″Not After″ date. Therefore, certificates may be created prior to their validity date and expire at some date in the future. It is common for certificates to have a life span of 3 months to 5 years.

The subject distinguished name specifies the certificate owner by using a specialized naming format known as a Distinguished Name (DN). A DN allows for the specification of the country, organization, city, state, owner name, and other attributes associated with the requesting entity (usually a person, but not limited to a person). The subject alternate name email allows for the specification of the owner's email address and the subject alternate name URI allows for the specification of the owner's Web site URI/URL.

Certificate authorities and certificates

Certificate Authorities (CA) issue, store, and typically publish certificates. A common place to publish certificates is on an LDAP server, because LDAP allows for easy access to community oriented data. Certificate Authorities also handle the revocation of certificates and the management of certificate revocation lists (CRLs). Revoking a certificate is the act of publishing the fact that a specific certificate is no longer valid due to reasons other than the expiration of the certificate's validity period. Because copies of certificates can be maintained and used outside the control of the issuing CA, CAs publish a list of revoked certificates in a CRL so that outside entities may query the list. This places the responsibility on entities using a copied certificate to compare the copied certificate against the issuing CA's CRL. A CA may only revoke certificates that it creates or issues. It cannot revoke certificates issued by other CAs.

Administrative reasons for revoking a certificate include:

- Compromise of the certificate's private key.
- Certificate owner left the company.
- Compromise of the CA.

CAs also have their own identifying certificate. This allows CAs to identify each other in peer-to-peer communications among other uses (for example, chains of trust).

Many CAs support the Certificate Management Protocol (CMP) for requesting and revoking certificates. The protocol supports multiple methods to establish a secure connection between a client (also known as an End Entity) and the CA, though not all clients and CAs support all methods. One common method requires each certificate creation and revocation request to use a reference number and password recognized by the CA. Other data such as a special certificate recognized by the CA may also be required. Revocation requests may require the matching private key of the certificate being revoked.

Although CMP provides for certificate creation and revocation requests, it does not support CRL query requests. In fact, CRLs are often accessed through out-of-band methods. Since CRLs are often published on LDAP servers, software applications can obtain the CRL from an LDAP server and manually scan the CRL. Another emerging method is the Online Certification Status Protocol (OCSP), but not all CAs support OCSP.

CAs are typically owned and operated by government organizations or trusted private organizations that attempt to provide assurance that certificates issued by them correspond to the person who requested the issuance of the certificate. The phrase *issuing a certificate* means to create a certificate and is not the same as requesting a copy of a published certificate.

Certificate storage format

The most common format for storing individual certificates is in Abstract Syntax Notation version 1 (ASN.1) format using the Distinguished Encoding Rules (DER). This format is referred to as *DER format*.

Keystores

A keystore (sometimes called a *keyset*) contains a user's private keys matching the public keys of their certificates.

A unique key label is assigned to every private key, usually by the user, for easy identification. Keystores are password-protected requiring a user to enter a password prior to accessing the keys or adding new keys. And typically, users have their own keystores. Keystores come in many different forms, for example: smart cards, LDAP-based, file-based, and so on. Not only do the forms vary, but so do the methods used to access them and the formats used to store the private key data. Certificate authentication service only supports file-based keystores.

Certificate Authentication Service implementation

The server side of certificate authentication service publishes certificates and certificate revocation lists (CRLs) that it creates to an LDAP server. The client side of certificate authentication service implements the user authentication, user administration, and user certificate management functions of certificate authentication service.

Certificate authentication service functions as a client/server model. The server side contains a Certificate Authority (CA) for creating and maintaining X.509 version 3 certificates and CRLs. (Typically, an organization uses one CA for the entire organization.) The client side contains the software (commands, libraries, load modules, and configuration files) required by every system participating in PKI authentication. The installation package for the server is **cas.server** and the installation package for the client is **cas.client**.

Creating PKI user accounts

To create a PKI user account, use the AIX **mkuser** command.

After it is created, each account has a certificate and a private keystore. (Existing accounts can be converted to PKI accounts too, but other steps are required.) The administrator supplies the keystore passwords to the new users, and new users can then log in to the system and change their keystore password.

User authentication data flow:

Users can have multiple certificates associated with their accounts. Each certificate has a unique, user defined tag value associated with it for easy identification, but only one certificate can be specified as the authentication certificate. Certificate authentication service uses a per-user attribute named *auth_cert* to specify which of the user's certificates is the user's authentication certificate. The value of the *auth_cert* attribute is the certificate's tag value.

The certificates, tags, matching keystore locations, matching key labels, and other related data are maintained under LDAP on a per-user basis. The combination of the user name and tag allows certificate authentication service to locate the certificate under the LDAP server. For more information on the PKI LDAP layer, see “PKI LDAP Layer (certificate storage)” on page 90.

At login, users supply a user name and password. Using the user name, the system retrieves the user’s authentication certificate tag from the user’s *auth_cert* attribute. Combining the user name and tag, the system retrieves the user’s certificate, keystore location, and matching key label from LDAP. It checks the validity period values found in the certificate to determine if the certificate has expired or has not reached its activation date. The system then retrieves the user’s private key by using the keystore location, key label, and supplied password. After the private key is retrieved, the system verifies that the private key and certificate match using an internal signing process. If the two match, the user passes the PKI authentication step of the login procedure. (This does not imply that the user is logged in. Several other account checks are performed by the AIX system on a user account before allowing the user access to the system.)

For a certificate to be used as an authentication certificate, the certificate must be signed using a trusted signing key. The signature is stored under LDAP with the certificate for later reference. The implementation requires that a certificate have a signature before the tag can be assigned to *auth_cert*.

The authentication process does not compare a certificate against a CRL. This is due to performance reasons (CRLs take time to acquire and scan and may be temporarily unavailable), but also due to publishing delays of CRLs (CAs may delay an hour or more before publishing a revoked certificate through a CRL, making certificate revocation a poor substitute for disabling a user’s account).

It is also worth noting that authentication does not require a CA. The majority of the work is performed locally by certificate authentication service, with the exception of retrieving data stored under LDAP.

Server implementation

The server side of certificate authentication service implements a CA written in Java and contains a Registration Authority (RA) along with self-auditing features. It publishes certificates and CRLs that it creates to an LDAP server.

The CA is configurable through a set of configuration files (Java property files). It contains an administrative application called **runpki** that provides sub-commands to start and stop the server among other functions and supports CMP for creating and revoking certificates. The CA requires Java 1.3.1, the IBM DB2® 7.1 database, and the IBM Directory 4.1. Due to DB2 requirements, the CA must run under a user account other than the root user.

The server contains the following commands to help install and manage the **cas.server** component:

mksecpki

This command is used during installation to configure the AIX PKI server components. As part of its tasks, the command creates a certificate authority user account for the certificate authority.

runpki

This command allows the system administrator to start the server. If the JavaPKI daemons are running, they must first be stopped. The **runpki** command starts the daemon in the background by using the *lb* flags combination. If the daemons need to be started in interactive mode, the administrator can edit the **runpki** command and use the *l* flag instead of the *lb* flags.

The **runpki** command must be run after performing an **su -** operation to the user account under which the certificate authority is running. The command is located in the **javapki** directory under the certificate authority user account’s home directory. (The **mksecpki** command creates the certificate authority user account.)

For example, if the certificate authority user account is pkiinst, then with root authority, type the following:

1. `su - pkiinst`
2. `cd javapki`
3. `runpki`

Client implementation

The client side of certificate authentication service implements the user authentication, user administration, and user certificate management functions of certificate authentication service.

After it is installed and configured on a system, certificate authentication service integrates into the existing user authentication and administration functions (such as the **mkuser**, **chuser**, **passwd**, and **login** commands) through the use of the AIX Loadable Authentication Module Framework (LAMF). It also adds several commands, libraries, and configuration files to help manage user certificates and keystores.

Certificate authentication service can be used in conjunction with either the AIX LDAP database mechanism or the file-based database mechanism for storing standard AIX attributes. Certificate authentication service always uses LDAP to maintain user certificates, even when the file-based database mechanism is used. For limitations when using the file-based database, see “Certificate Authentication Service planning” on page 98.

The client side of certificate authentication service contains the most user oriented software of the two parts. For this reason, the following sections describe how certificate authentication service maintains and uses the data required for PKI authentication.

General client features:

This section describes some of the general features of certificate authentication service.

These features include:

- Provides user authentication via PKI certificates
- Provides commands to manage user certificates and keystores
- Supports multiple certificates per user
- Supports multiple CA’s simultaneously
- Integrates into existing AIX administration commands and authentication (for example, **login**, **passwd**, **mkuser**)
- Generate certificates at user creation time or add certificates after user creation
- Works with either an LDAP user database or the standard AIX file-based user database
- Configurable key sizes and algorithms
- Associates certificates with Process Authentication Groups (PAGs).

General client architecture:

The client architecture of the Certificate Authentication Service takes a layered approach.

Java daemon:

At the foundation of the client side is a Java-based daemon using the JCE security package.

The Java daemon manages user keystores, creates key pairs, performs CMP communications, and provides all hashing and encryption functions. Because APIs of PKI service provider packages are not standardized for C applications, a wrapper layer API called the Service Management Layer (SML) provides a normalized API to application programs and daemons.

Service Management Layer:

Service Management Layer creates certificates and keystores, and manages keystores, but it doesn't manage certificate storage.

The SML service for the Java daemon is named **/usr/lib/security/pki/JSML.sml**. Certificate storage is managed by the PKI LDAP Layer.

Private Key Storage Through SML

The Java daemon uses PKCS#12 formatted keystore files for storing user keys. The keystores are protected by a single password used to encrypt all the keys in the keystore. The location of a keystore is specified as a URI. By default, certificate authentication service maintains keystore files in the **/var/pki/security/keys** directory.

Keystores are typically limited in size, including file keystores. The SML Layer provides the API for managing keystores.

Certificate authentication service supports only file keystores. It does not support smart card or LDAP keystores. You can support roaming users by placing the file keystores on a shared file system under the same mount point on all systems.

PKI LDAP Layer (certificate storage):

Certificate authentication service stores certificates and other certificate related information on a per-user basis in LDAP through the PKI LDAP Layer.

A user account can have multiple certificates associated with it. Each association has a unique, user-specified tag for easy identification and lookup. Certificate authentication service uses the combination of the user's name and the tag to locate a user's certificate association in LDAP.

For performance versus disk space trade-offs, certificate authentication service can save either the entire certificate under LDAP or just a URI reference to the certificate. If a URI reference is used instead of a certificate, certificate authentication service queries the reference to obtain the actual certificate. References are most commonly used in conjunction with a CA which publishes its certificates on an LDAP sever. The types of URI references currently supported by certificate authentication service are LDAP references. Certificate authentication service stores certificates in DER format and expects URI references to refer to DER formatted certificates.

Certificate authentication service also stores the type and location of each certificate's matching keystore and key label in the same record as the certificate association on the LDAP server. This allows users to have more than one keystore and allows certificate authentication service to quickly find a certificate's matching private key. To support roaming users, a user's keystore must reside in the same location on all systems.

Certificate authentication service maintains the *auth_cert* attribute in LDAP on a per-user basis. This attribute specifies the tag of the certificate used for authentication.

All LDAP information is readable by ordinary users, except for the *auth_cert* attribute which is restricted to the LDAP **ldappkiadmin** account. Since the root user has access to the LDAP **ldappkiadmin** password through the **acct.cfg** file, applications running with the effective UID of root can access the *auth_cert* attribute. (This applies to the accessibility of the URI reference value, not to the data referenced by the URI reference value. Typically, the data referenced by the URI reference value is public.) The API for managing the certificate storage is contained in the **libpki.a** library.

libpki.a library:

In addition to serving as the home of the SML APIs and the PKI LDAP Layer APIs, the **libpki.a** library houses several subroutines.

The library includes APIs that do the following:

- Manage the new configuration files
- Access certificate specific attributes
- Combine multiple lower layer functions into higher level functions
- Are expected to be common among SML services

Note: The APIs are not published.

Loadable Authentication Module Framework Layer:

On top of the SML API and PKI LDAP API resides the Loadable Authentication Module Framework (LAMF) layer. LAMF supplies AIX authentication and user administration applications with common authentication and user administration APIs regardless of the underlying mechanism (for example, Kerberos, LDAP, DCE, files).

LAMF uses the SML API and the PKI LDAP API as building blocks in implementing PKI authentication. It does this through the use of load modules that map LAMF's API to different authentication/database technologies. Commands like **login**, **telnet**, **passwd**, **mkuser**, and others use the LAMF API to implement their functions; hence, these commands automatically support new authentication and database technologies when new load modules for these technologies are added to the system.

Certificate authentication service adds a new LAMF load module to the system named **/usr/lib/security/PKI**. The module must be added by the system administrator to the **/usr/lib/security/methods.cfg** file before using PKI for authentication. The module must also be paired with a database type (for example, LDAP) in the **methods.cfg** file before it can be used for authentication. An example of the **methods.cfg** file containing the LAMF module and database definition can be found in "methods.cfg file" on page 110.

Once the definitions are added to **methods.cfg**, the administrator can set the *registry* and *SYSTEM* user attributes (defined in the **/etc/security/user** file) to the new stanza value or values for PKI authentication.

Client commands:

Above all the API layers (LAMF, PKI LDAP, and SML) reside the commands.

Besides the standard AIX authentication and user administration commands supporting certificate authentication service (through LAMF), several certificate authentication service specific commands exist. These commands help the user manage certificates and keystores. Below is a list of the commands along with a brief description.

certadd
Adds a certificate to the user's account in LDAP and checks if the certificate is revoked.

certcreate
Creates a certificate.

certdelete
Deletes a certificate from the user's account (i.e., from LDAP).

certget
Retrieves a certificate from the user's account (i.e., from LDAP).

certlink
Adds a link to a certificate that exists in a remote repository to the user's account in LDAP and checks if the certificate is revoked.

certlist
Lists the certificates associated with the user's account contained in LDAP.

certrevoke
Revokes a certificate.

certverify
Verifies the private key matches the certificate and performs trusted signing.

keyadd
Adds a keystore object to a keystore.

keydelete
Deletes a keystore object from a keystore.

keylist
Lists the objects in a keystore.

keypasswd
Changes the password on a keystore.

For more information about these commands. see the *AIX 5L Version 5.3 Commands Reference*.

Process Authentication Group commands:

The Process Authentication Group (PAG) commands are new to AIX. PAGs are data items that associate user-authentication data with processes.

For certificate authentication service, if the PAG mechanism is enabled, the user's authentication certificate is associated with the user's login shell. As the shell creates child processes, the PAG propagates to each child.

The PAG mechanism requires the **/usr/sbin/certdaemon** daemon to be enabled in order to provide this functionality. By default, the mechanism is not enabled. Certificate authentication service does not require the PAG mechanism to be enabled, but works with the mechanism if it is enabled.

To enable the **certdaemon** daemon, add the following line to the **/etc/inittab** file:

```
certdaemon:2:wait:/usr/sbin/certdaemon
```

A list of PAG commands along with brief descriptions follows:

paginit
Authenticates a user and creates a PAG association.

pagdel
Lists authentication information associated with the current process.

paglist
Removes existing PAG associations within the current process' credentials.

For more information about these commands, see the *AIX 5L Version 5.3 Commands Reference*.

User administration commands:

Similar to user-authentication, certificate authentication service integrates with the AIX user-administration functions through the AIX LAMF. Commands like **chuser**, **lsuser**, **mkuser**, and **passwd** use the LAMF API to implement their functions. Therefore, these commands automatically support new authentication and database technologies when new load modules for these technologies are added to the system.

The subsections below provide a more in-depth look at how PKI authentication affects the user administration commands.

The following commands are affected by the PKI authentication process:

chuser
This command allows the administrator to modify the *auth_cert* user attribute. This attribute specifies the tag value of the certificate used for authentication. The certificate must be signed by the trusted signing key in order to be used as the authentication certificate. (Certificate attributes, certificate storage attributes, and keystore attributes are not available through this command.)

lsuser This command lists the value of the user's *auth_cert* attribute, as well as, the certificate attributes listed below. The *auth_cert* attribute specifies the tag value of the certificate used for authentication. (Other certificate attributes, certificate storage attributes, and keystore attributes are not available through this command.)

The certificate attributes listed by the **lsuser** command are as follows:

subject-DN
The user's subject distinguished name.

subject-alt-name
The user's subject alternate name email.

valid-after
The date the user's certificate becomes valid.

valid-until
The date the user's certificate becomes invalid.

issuer The distinguished name of the issuer.

mkuser
This command provides an administrator the option of generating a certificate at user creation time. An administrator can use the **mkuser** command to generate a certificate during user creation for users who don't already have an authentication certificate. Optionally, if a user already has an authentication certificate, but no user account, the administrator can create the account without generating a certificate and add the certificate (and keystore) later. The default value for this option is specified in the **/usr/lib/security/pki/policy.cfg** file in the newuser stanza by the *cert* attribute.

Many default values are required when automatically generating an authentication certificate for a user using the **mkuser** command. Many of these values are specified in the **newuser** stanza of the **/usr/lib/security/pki/policy.cfg** file. The newuser stanza provides administrative control over these default values. Some of the default values are as follows:

- CA
- Value for the *auth_cert* attribute
- Location for the keystore
- Password for the keystore
- Private key label
- Domain name for the subject alternate name e-mail field

A behavioral difference between creating a PKI user account and a non-PKI user account is that creating a PKI user account requires a password to encrypt the private key if the **mkuser** command generates an authentication certificate for the account. Since the **mkuser** command is a non-interactive command, the command obtains the password from the **policy.cfg** file and sets the keystore password (the private key password) to this value; therefore, the account is immediately accessible after creation. When creating a non-PKI user account, the **mkuser** command sets the password to an invalid value, preventing accessibility.

passwd
This command modifies the user's keystore password when used on a PKI user account. It enforces the password restriction rules found in the **/etc/security/user** file, it enforces the flags attribute found in the **/etc/security/passwd** file, and it enforces any rules required by the PKI service provider.

Because file-based keystores encrypt their private keys using the user's password, the root user cannot reset a file-based keystore password without knowing the keystore's current password. If a user forgets their keystore password, the root user will not be able to reset the password unless root knows the keystore's password. If the password is unknown, a new keystore and new certificates may have to be issued to the user.

Configuration files:

Certificate authentication service uses configuration files for configuring the client-side: **acct.cfg**, **ca.cfg**, and **policy.cfg**.

The SMIT interface provides support for these configuration files. The following sections provide information about the configuration files.

acct.cfg file

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, such as CMP reference numbers and passwords. The LDAP stanzas contain private LDAP information not suitable for public access, such as PKI LDAP administrative names and passwords.

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. The LDAP stanzas are all named `ldap`, and 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 also exist.

CA stanzas contain the following attributes:

capasswd
Specifies the CA's CMP password. The length of the password is specified by the CA.

carefnum
Specifies the CA's CMP reference number.

keylabel
Specifies the label of the private key in the trusted keystore used to sign certificate requests.

keypasswd
Specifies the password for the trusted keystore.

rvpasswd
Specifies the revocation password used for CMP. The length of the password is specified by the CA.

rvrefnum
Specifies the revocation reference number used for CMP.

The LDAP stanza contains the following attributes:

ldappkiadmin
Specifies the account name of the LDAP server listed in **ldapservers**.

ldappkiadmpwd
Specifies the password for the LDAP server's account.

ldapservers
Specifies the LDAP server name.

ldapsuffix
Specifies the DN attributes added to a user's certificate DN by the **mkuser** command.

The following is an example **acct.cfg** file:

```
local:
 carefnum = 12345678
 capasswd = password1234
 rvrefnum = 9478371
 rvpasswd = password4321
 keylabel = "Trusted Key"
 keypasswd = joshua

ldap:
 ldappkiadmin = "cn=admin"
 ldappkiadmpwd = secret
 ldapservers = "LDAP server.austin.ibm.com"
 ldapsuffix = "ou=aix,cn=us"
```

For more information, see the *AIX 5L Version 5.3 Files Reference*.

ca.cfg file

The **ca.cfg** file consists of CA stanzas. The CA stanzas contain public CA information used by certificate authentication service for generating certificate requests and certificate revocation requests.

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 stanzas should be named `ldap` or `default`.

CA stanzas contain the following attributes:

algorithm
Specifies the public key algorithm (for example, RSA).

crl Specifies the CA's CRL URI.

dn Specifies the base DN used when creating certificates.

keysize
Specifies the minimum key size in bits.

program
Specifies the PKI service module file name.

retries
Specifies the number of retry attempts when contacting the CA.

server Specifies the CA's URI.

signinghash
Specifies the hash algorithm used to sign certificates (for example, MD5).

trustedkey
Specifies the trusted keystore containing the trusted signing key used for signing authentication certificates.

url Specifies the default value for the subject alternate name URI.

The default CA stanza is named local. The following is an example **ca.cfg** file:

```
local:
 program = /usr/lib/security/pki/JSML.sml
 trustedkey = file:/usr/lib/security/pki/trusted.p15
 server = "cmp://9.53.230.186:1077"
 crl = "ldap://dracula.austin.ibm.com/o=aix,c=us"
 dn = "o=aix,c=us"
 url = "http://www.ibm.com/"
 algorithm = RSA
 keysize = 512
 retries = 5
 signinghash = MD5
```

For more information, see the *AIX 5L Version 5.3 Files Reference*.

policy.cfg file

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** commands use the comm and crl stanzas.

The newuser stanza contains the following attributes:

ca Specifies the CA used by the **mkuser** command when generating a certificate.

cert Specifies whether the **mkuser** command generates a certificate (new) or not (get) by default.

domain
Specifies the domain part of the certificate's subject alternate name e-mail value 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.

keystore
Specifies the keystore URI 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.

label Specifies the private key label used by the **mkuser** command when generating a certificate.

passwd
Specifies the keystore's password 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.

tag Specifies the *auth_cert* tag value used by the **mkuser** command when creating a user when cert = new.

validity
Specifies the certificate's validity period value 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.

The storage stanza contains the following attributes:

replicate
Specifies whether the **certlink** command saves a copy of the certificate (yes) or just the link (no).

The crl stanza contains the **check** attribute, which specifies whether the **certadd** and **certlink** commands should check the CRL (yes) or not (no).

The comm stanza contains the **timeout** attribute which specifies the timeout period in seconds used by **certadd** and **certlink** when requesting certificate information using HTTP (for example, retrieving CRLs).

The following is an example of the **policy.cfg** file:

```
newuser:
 cert = new
 ca = local
 passwd = pki
 version = "3"
 keysize = 512
 keystore = "file:/var/pki/security/keys"
 validity = 86400

storage:
 replicate = no

crl:
 check = yes

comm:
 timeout = 10
```

For more information, see the *AIX 5L Version 5.3 Files Reference*.

Audit-log events:

The Certificate Authentication Service (CAS) client generates several audit-log events.

- CERT_Create
- CERT_Add
- CERT_Link
- CERT_Delete
- CERT_Get
- CERT_List
- CERT_Revoke
- CERT_Verify
- KEY_Password
- KEY_List
- KEY_Add
- KEY_Delete

Trace events:

The Certificate Authentication Service (CAS) client generates trace events.

The CAS client generates several new trace events in the 3B7 and 3B8 range.

Certificate Authentication Service planning

Certificate Authentication Service (CAS) is available beginning with AIX 5.2. The minimum software requirements for CAS are a DB2 server, an IBM Directory server, and a certificate authentication service server. All can be installed on one system or on a combination of systems. Each enterprise must determine the best choice for their environment.

This section provides information on planning for certificate authentication service, as follows:

- “Certificate considerations”
- “Keystore considerations”
- “User registry considerations”
- “Configuration considerations” on page 99
- “Security considerations” on page 99
- “Other Certificate Authentication Service considerations” on page 100

Certificate considerations

Certificate authentication service supports X.509 version 3 certificates. It also supports several version 3 certificate attributes, but not all certificate attributes.

For a list of supported certificate attributes, see the **certcreate** command and the **ca.cfg** file. Certificate authentication service contains limited support of the Teletex character set. Specifically, only 7-bit (ASCII subset of) Teletex is supported by certificate authentication service.

Keystore considerations

Certificate authentication service supports keystore files. Smart cards, LDAP keystores, and other types of keystores are not supported.

By default, user keystores are kept in the local file system under the **/var/pki/security/keys** directory. Because the keystores are local to the system, they cannot be accessed by other systems; thus, user authentication will be restricted to the system containing the user’s keystore. To allow for roaming users, either copy the user’s keystore to the identical location with the same keystore name on other systems or place the keystores on a distributed file system.

Note: Care must be taken to ensure that access permission to the user’s keystore remains unchanged. (In AIX, every certificate in LDAP contains the path name to the private keystore containing the certificate’s private key. The keystore must exist at the path name specified in LDAP in order to be used for authentication.)

User registry considerations

Certificate authentication service supports an LDAP user-registry. LDAP is also the recommended user registry type to use with certificate authentication service.

Certificate authentication service also supports a file-based user registry. Certain restrictions must be enforced by the administrator for file-based PKI to work correctly. Specifically, identically named user accounts on different systems participating in PKI authentication must refer to the same account.

For example, user *Bob* on *system A* and user *Bob* on *system B* must refer to the same user *Bob*. This is because certificate authentication service uses LDAP to store certificate information on a per user basis. The user name is used as the indexing key to access this information. Because file-based registries are local to each system and LDAP is global to all systems, the user names on all systems participating in PKI authentication must map to unique user names in the LDAP namespace. If user *Bob* on *system A* is different from user *Bob* on *system B*, either only one of the *Bob*’s can participate in PKI authentication or each *Bob* account must use a different LDAP namespace/server.

Configuration considerations

For configuration simplicity, consider maintaining the three configuration files (**acct.cfg**, **ca.cfg**, and **policy.cfg**) on a distributed file system using symbolic links to avoid having to modify configuration files on every system.

Maintain proper access-control settings on these files. This situation may increase your security vulnerability because the information in these files will be transferred across your network.

Security considerations

The **acct.cfg** and **ca.cfg** files contain sensitive reference numbers, passwords, and certificate information.

acct.cfg file

The **acct.cfg** file contains sensitive CA reference numbers and passwords (see the **carefnum**, **capasswd**, **rvrefnum**, and **rvpasswd** attribute descriptions for **acct.cfg**). These values are used solely for CMP communications with the CA when creating a certificate and revoking a certificate, respectively. If compromised, the compromiser may be able to create certificates at will, and revoke anyone's certificate at will.

To limit the exposure, consider restricting certificate creation or revocation to a small number of systems. The **carefnum** and **capasswd** attribute values are required only on systems where certificates are created (either through the **certcreate** or **mkuser** commands). This may imply limiting user account creation to the same set of systems.

Note: The **mkuser** command can be configured to automatically create a certificate during user creation or it can create an account without a certificate, whereby the administrator must create and add the certificate at a later time.

Similarly, the **rvrefnum** and **rvpasswd** attribute values are required only on systems where certificates are to be revoked (through the **certrevoke** command).

The **acct.cfg** file also contains sensitive trusted signing key information (see the **keylabel** and **keypasswd** attribute descriptions for the **acct.cfg** file). These values are used solely for special certificate verification operations. If compromised, the compromiser may be able to forge verified certificates.

To limit the exposure, consider restricting certificate verification to a small number of systems. The **keylabel** and **keypasswd** attribute values of the **acct.cfg** file and the **trustedkey** attribute value of the **ca.cfg** file are required only on systems where certificate verification is required. Specifically, on systems where the **mkuser** (with automatic certificate creation enabled) and **certverify** commands are required.

Active new accounts

When creating a PKI user account, if the **cert** attribute of the newuser stanza in the **policy.cfg** file is set to `new`, the **mkuser** command creates an active PKI account complete with a working certificate and password. The password on the account is specified by the **passwd** attribute in the newuser stanza. Because keystores require a password in order to store private keys. This differs from other types of user account creations where the administrator must first create the account, then set the password before the account is activated.

The root user and keystore passwords

Unlike other account types where the root user can change an account's password without knowing the account's password, PKI accounts do not allow this. This is because account passwords are used to encrypt keystores and keystores cannot be decrypted without knowing the password. When users forget their passwords, new certificates must be issued and new keystores created.

Other Certificate Authentication Service considerations

When planning for the Certificate Authentication Service (CAS), include the following:

- Certificate authentication service contains its own certificate authority (CA). Other CA implementations are not supported by certificate authentication service.
- The larger the key size, the more time required to generate key pairs and to encrypt data. Hardware based encryption is not supported.
- Certificate authentication service uses the IBM Directory for LDAP. Other LDAP implementations are not supported by certificate authentication service.
- Certificate authentication service uses DB2 for database support. Other database implementations are not supported by certificate authentication service.
- Certificate authentication service requires all commands, libraries, and daemons run in a Unicode environment.

Packaging of Certificate Authentication Service

This section shows the package components of the Certificate Authentication Service (CAS).

Table 8. Packaging of Certificate Authentication Service

Package Name	Fileset	Contents	Dependencies	Installation
cas.server	cas.server.rte	Certificate Authority (CA)	• AIX 5.2 • Java131 (ships with AIX base media) • Java131 Security Extensions (ships with Expansion Pack) • IBM Directory Server (LDAP) • DB2 7.1	Manual
cas.client	cas.client.rte	• Cert commands • PKI Auth Load Module • libpki.a • SML Module • Config Files • Java Daemon	• AIX 5.2 • Java131 (ships with AIX base media) • Java131 Security Extensions (ships with Expansion Pack) • IBM Directory Client (LDAP) • PAG (assumed)	Manual
cas.msg	cas.msg.[lang].client	Message catalogs	cas.client	Manual
bos	bos.security.rte	PAG commands and daemon	not applicable	Installed with kernel

The **cas.server** package contains the CA and installs in the **/usr/cas/server** and **/usr/cas/client** directories. An organization typically uses only one CA, and therefore, this package is installed manually. This package prerequisites the IBM Directory server side, **db2_07_01.client**, **Java131.rte**, and **Java131.ext.security**. The **Java131.rte package** is installed by default when the AIX 5.2 operating system is installed, but the other packages are manually installed.

In order for the **db2_07_01.client** package to work, the **db2_07_01.server** package must be installed on a system that is on the network.

The **cas.client** package contains the files required for every client system supporting certificate authentication service. Without this package, a system cannot participate in AIX PKI authentication.

Certificate Authentication Service installation and configuration

This section lists the procedures to install and configure the Certificate Authentication Services (CAS).

LDAP server for PKI installation and configuration

This section shows some scenarios that are possible when installing and configuring LDAP for PKI user certificate data.

Installing the LDAP server:

Detailed instructions for installing the IBM Directory Server software can be found in the product documentation contained in the **ldap.html.en_US.config** fileset. After installing the **ldap.html.en_US.config** fileset, the documentation can be viewed using a Web browser at the following URL: **file:/usr/ldap/web/C/getting_started.htm**.

Perform the following steps to install the LDAP server:

1. Login as the **root** user.
2. Place volume 1 of the AIX Base Operating System CDs in the CD-ROM drive.
3. Type `smitty install_latest` at the command line and press Enter.
4. Select **Install Software**.
5. Select the input device or software directory containing the IBM Directory Server software and press Enter.
6. Use the **F4** key to list the install packages in the **Software to Install** field.
7. Select the LDAP server package and press Enter.
8. Verify that the **AUTOMATICALLY install requisite software** option is set to **YES**, and press Enter. This will install the LDAP server and client filesets and the DB2 backend database filesets.

   The filesets installed include the following:

   - LDAP client **.adt** (Directory Client SDK)
   - LDAP client **.dmt** (Directory Client DMT)
   - LDAP client **.java** (Directory Client Java)
   - LDAP client **.rte** (Directory Client Run-time Environment)
   - LDAP server **.rte** (Directory Server Run-time Environment)
   - LDAP server **.admin** (Directory Server)
   - LDAP server **.cfg** (Directory Server Config)
   - LDAP server **.com** (Directory Server Framework)
   - **db2_07_01.*** (DB2 Run-time Environment and associated filesets)
9. Install the DB2 package, **db2_07_01.jdbc**. The DB2 package, **db2_07_01.jdbc**, is located on the Expansion Pack CD. Use the installation procedure listed above to install the **db2_07_01.jdbc** package.

Configuring the LDAP server:

After the LDAP and DB2 filesets have been installed, the LDAP server must be configured.

Even though the configuration can be done through the command line and file editing, for ease of administration and configuration, the LDAP web administrator is used. This tool requires a web server.

The Apache web server application is located on the AIX Toolbox for LINUX Applications CD. Use either the SMIT interface or the **geninstall** command to install the Apache web server. Other web servers can also be used, see the LDAP documentation for details.

You can find detailed instructions for configuring LDAP in the product HTML documentation. To configure the LDAP, perform the following steps:

1. Use **ldapcfg** to set the admin DN and password for the LDAP database. The administrator is the **root** user of the LDAP database. To configure an administrator DN of **cn=admin** with a password of **secret**, type the following:

   ```
   # ldapcfg -u cn=admin -p secret
   ```

   The DN and password will be required later when configuring each client. Specifically, the DN and password will be used as the *ldappkiadmin* and *ldappkiadmpwd* attributes of an **ldap** stanza in the **acct.cfg** file.
2. Configure the web administrator tool using the location of the web server configuration file, as follows:

   ```
   # ldapcfg -s apache -f /etc/apache/httpd.conf
   ```
3. Restart the web server. For the Apache server, use the command:

   ```
   # /usr/local/bin/apachectl restart
   ```
4. Access the web administrator using the URL **http:// hostname/ldap**. Then login using the LDAP administrator DN and password configured in step 2.
5. Using the web administrator tool, follow the directions to configure the DB2 database backend and restart the LDAP server.

Configuring the LDAP Server for PKI:

Certificate authentication service requires two separate LDAP directory information trees. One tree is used by the CA for publishing certificates and CRLs. The other tree is used by each client for storing and retrieving per-user PKI data.

The following steps configure the LDAP directory information tree used for storing and retrieving per-user PKI data.

1. **Add the LDAP Configuration Suffix Entry**. The default suffix for the PKI data is **cn=aixdata**. This places the PKI certificate data below the default suffix for all AIX data. The default data root for the PKI data is **ou=pkidata,cn=aixdata**. All PKI data is placed under this location.

   PKI Data Suffix

   cn=aixdata

   Common suffix for all AIX data. May already exist if LDAP server is being used for other AIX data.

   The suffix configuration entry can be added through the web administrator tool, or by directly editing the LDAP server configuration file.

   To add the suffix configuration entry using the Web administrator, do the following:

   a. Select **Settings** from the left side menu.
   b. Select **Suffixes**.
   c. Enter the necessary suffix for the PKI data, and then click the **Update** button.
   d. Restart the LDAP server, after the suffix is successfully added.

   To add the suffix configuration entry by editing the LDAP server configuration file, do the following:

   a. In the **/usr/ldap/etc/slapd32.conf** file, locate the line containing

      ```
      ibm-slapdSuffix: cn=localhost
      ```

      This is the default system suffix.
   b. Add the necessary `ibm-slapdSuffix` entry for the PKI data. For example, you can add a suffix entry similar to the following:

      ```
      ibm-slapdSuffix: cn=aixdata
      ```
   c. Save the configuration file changes.

d. Restart the LDAP server.

2. **Add the PKI Data Suffix, Root, and ACL Database Entries**. The Data Root is the point in the LDAP directory structure under which all the PKI data resides. The ACL is the Access Control List for the Data Root that sets the access rules for all the PKI data. The **pkiconfig.ldif** file is supplied to add the suffix, root, and ACL entries to the database.

   a. First, add the suffix and root database entries and the PKI data administrator password. The first part of the file adds the default suffix entries to the database and sets the password as follows:

   ```
   dn: cn=aixdata
   objectclass: top
   objectclass: container
   cn: aixdata

   dn: ou=pkidata,cn=aixdata
   objectclass: organizationalUnit
   ou: cert
   userPassword: <<password>>
   ```

   b. Edit the **pkiconfig.ldif** file and replace the <<password>> character string after the **userPassword** attribute with your password for the PKI data administrator.

   The DN and **userPassword** values will be required later when configuring each client. Specifically, the DN (`ou=pkidata,cn=aixdata`) and value for *password* will be used as the **ldappkiadmin** and **ldappkiadmpwd** attributes of an ldap stanza in the **acct.cfg** file.

   The second part of the file changes the ownership and adds the ACL for the PKI data as follows:

   ```
   dn: ou=pkidata,cn=aixdata
   changetype: modify
   add: entryOwner
   entryOwner: access-id:ou=pkidata,cn=aixdata
   ownerPropagate: true

   dn: ou=pkidata,cn=aixdata
   changetype: modify
   add: aclEntry
   aclEntry: group:cn=anybody:normal:grant:rsc:normal:deny:w
   aclEntry: group:cn=anybody:sensitive:grant:rsc:sensitive:deny:w
   aclEntry: group:cn=anybody:critical:grant:rsc:critical:deny:w
   aclEntry: group:cn=anybody:object:deny:ad aclPropagate: true
   ```

   Note: To avoid jeopardizing the integrity of your PKI implementation, *do not* make any changes to the ACL settings.

   The **pkiconfig.ldif** file can be edited to use a suffix other than the default, however this is recommended only for experienced LDAP administrators. The **ldif** file can then be applied to the database using the **ldapadd** command below.

   c. Replace the values for the **-D** and **-w** options with your local LDAP administrator DN and password, as follows:

   ```
   # ldapadd -c -D cn=admin -w secret -f pkiconfig.ldif
   ```

3. **Restart the LDAP Server**. Restart the LDAP server using the web administrator tool, or by killing and restarting the **slapd** process.

Installing and configuring the Certificate Authentication Service

This section provides steps for installing and configuring the certificate authentication service.

To install and configure the certificate authentication service, do the following:

1. Install the Java security filesets (**Java131.ext.security.***) from the Expansion Pack CD. The required packages are as follows:
   - **Java131.ext.security.cmp-us** (Java Certificate Management)
   - **Java131.ext.security.jce-us** (Java Cryptography Extension)

   - **Java131.ext.security.jsse-us** (Java Secure Socket Extension)
   - **Java131.ext.security.pkcs-us** (Java Public Key Cryptography)
2. Move the **ibmjcaprovider.jar** file from **/usr/java131/jre/lib/ext** to another directory. This file conflicts with the Java security filesets and must be moved for correct functioning of the certificate authentication service.
3. Install the certificate authentication service server fileset (**cas.server.rte**) from the Expansion Pack CD.

Configuring the Certificate Authentication Service server to work with LDAP

To configure the Certificate Authentication Service (CAS) server to work with LDAP, perform these steps:

1. If not already installed, then install the IBM Directory client package on the system supporting the **cas.server** package.
2. If not already configured, then configure the IBM Directory client, as follows:

   ```
   # ldapcfg -l /home/ldapdb2 -u "cn=admin" -p secret -s apache \
             -f /usr/local/apache/conf/httpd.conf
   ```

   It is assumed that the Web Server is the Apache Web Server in the above configuration command.
3. Add the following suffix to the **slapd.conf** file, as follows:

   ```
   ibm-slapdSuffix: o=aix,c=us
   ```

   You can specify a different distinguished name instead of `o=aix,c=us`.
4. Run the **slapd** command, as follows:

   ```
   # /usr/bin/slapd -f /etc/slapd32.conf
   ```
5. Add the object classes, as follows:

   ```
   # ldapmodify -D cn=admin -w secret -f setup.ldif
   ```

   where **setup.ldif** contains the following:

   ```
   dn: cn=schema
   changetype: modify
   add: objectClasses
   objectClasses: ( 2.5.6.21 NAME 'pkiuser' DESC 'auxiliary class for non-CA certificate owners'
     SUP top AUXILIARY MAY userCertificate )

   dn: cn=schema
   changetype: modify
   add: objectClasses
   objectClasses: ( 2.5.6.22 NAME 'pkiCA' DESC 'class for Cartification Authorities' SUP top
     AUXILIARY MAY ( authorityRevocationList $ caCertificate $ certificateRevocationList $
     crossCertificatePair ) )

   dn:cn=schema
   changetype: modify
   replace: attributetypes
   attributetypes: ( 2.5.4.39 NAME ( 'certificateRevocationList'
     'certificateRevocationList;binary' ) DESC '  ' SYNTAX 1.3.6.1.4.1.1466.115.121.1.5
     SINGLE-VALUE )

   replace:ibmattributetypes
   ibmattributetypes:( 2.5.4.39 DBNAME ( 'certRevocationLst' 'certRevocationLst' )
     ACCESS-CLASS NORMAL)
   ```
6. Add the entries:

   ```
   # ldapadd -D cn=admin -w secret -f addentries.ldif
   ```

   where *addentries.ldif* contains the following:

   ```
   dn: o=aix,c=us
   changetype: add
   objectclass: organization
   objectclass: top
   objectclass: pkiCA
   o: aix
   ```

Note: Sample **addentries.ldif** and **setup.ldif** files are provided in the **cas.server** package.

7. Stop and start the **slapd** daemon.

Creating the Certificate Authority:

This section provides the steps to create the certificate authority.

1. Create a reference file. The reference file contains one or more certificate creation reference number and password pairs. A pair represents the authentication information accepted by the certificate authentication service server when a certificate authentication service client attempts to authenticate to the server during the creation of a certificate (typically using the CMP protocol). The format of the file is a reference number followed by a password, both on separate lines. For example:

   ```
   12345678
   password1234
   87654321
   password4321
   ```

   where `12345678` and `87654321` are reference numbers, and `password1234` and `password4321` are their respective passwords. Blank lines are not allowed. Space characters should not precede or follow reference numbers or passwords. At least one reference number and password must exist in the file. An example file can be found in **/usr/cas/server/iafile**. You will need to reference these values each time you set up a client.

2. Configure the CA using the **mksecpki** command as follows:

   ```
   # mksecpki -u pkiuser -f /usr/cas/server/iafile -p 1077 -H ldap.cert.mydomain.com \
             -D cn=admin -w secret -i o=aix,c=us
   ```

   Information on the **mksecpki** flags follows:

   -u Specifies a user account name where the certificate authentication service server will be installed.

   -f Specifies the reference file created in the previous step.

   -p Specifies a port number for the LDAP server.

   -H Specifies the LDAP server host name or IP address.

   -D Specifies the LDAP administrator's common name.

   -w Specifies the LDAP administration password.

   -i Specifies the LDAP branch where the user certificate data will reside.

   The **mksecpki** command automatically generates the trusted signing key with a key label of **TrustedKey**, the password of the CA user account, and places it in the **/usr/lib/security/pki/trusted.pkcs12** keystore file. It's not necessary to perform the steps in "Creating the trusted signing key" unless you need to generate multiple keys or want a trusted signing key with a different key label and/or password.

Creating the trusted signing key:

The **mksecpki** command automatically generates a trusted signing key with a key label of *TrustedKey*, the password of the CA user account, and places it in the **/usr/lib/security/pki/trusted.pkcs12** keystore file. If you need to generate a new trusted signing key or multiple trusted signing keys, then this section provides the steps needed to generate a trusted signing key.

All certificate authentication service clients where certificate creation and revocation are allowed require a trusted signing key for signing the user's authentication certificate. The key is saved in a separate keystore and is made available to all systems where certificates can be created. A single key can be used by all systems or, for a more secure approach, multiple keys can be created and distributed.

To create a trusted key, use the **/usr/java131/bin/keytool** command. Use a file name of a non-existing file. The **keytool** command prompts for a keystore password and key password. Both the keystore password and key password must be identical for certificate authentication service to access the key in the keystore. Run the **keytool** command as follows:

```
keytool -genkey -dname `cn=trusted key' -alias `TrustedKey' -keyalg RSA \
        -keystore filename.pkcs12 -storetype pkcs12ks
```

In this example, the trusted key label is *TrustedKey* and the trusted keystore password is user-supplied. Remember these values, because you will need them when configuring the certificate authentication service clients. When configuring a certificate authentication service client, the **keylabel** and **keypasswd** attributes in the **acct.cfg** file will need to be set to the trusted key label and trusted keystore password, respectively.

For security reasons, make sure the keystore file (*filename*.pkcs12) is read and write protected. Only the root user should have access to this file. The trusted key should be the only object in the keystore.

Configuring the Certificate Authentication Service client

There are many configuration options on the client side of Certificate Authentication Service. The following sections provide the configuration procedure required for each system participating in PKI authentication.

Trusted Signing Key installation

For information on creating the trusted signing key, see “Creating the trusted signing key” on page 105. The default location for the trusted keystore is in the **/usr/lib/security/pki** directory.

For security reasons, make sure the keystore file is read and write protected. Only the root user should have access to this file.

Editing the acct.cfg file

Remove any ldap stanzas that may exist in the **/usr/lib/security/pki/acct.cfg** file using a text-based editor like the **vi** command.

Certificate Authority account configuration:

Minimally, the local CA account must be configured. By default, the local CA account exists, but must be modified to match your environment.

Certificate authentication service supports the use of multiple CA’s by a single system through stanza-based configuration files. The default CA stanza name of `local` is used when a CA is not specified by a user or by the software. All systems must have a valid local stanza definition in the appropriate certificate authentication service configuration files. Only one CA may have a stanza name of `local`. All other CA’s must have a unique stanza name. CA stanza names cannot be `ldap` or `default`.

The following sections guide you through the SMIT configuration screens for configuring the local CA.

Change/Show a Certificate Authority:

To change/show a Certificate Authority (CA), perform these steps:

1. Run PKI SMIT, as follows:
   ```
   smitty pki
   ```
2. Select **Change / Show a Certificate Authority**.
3. Type `local` for the **Certificate Authority Name** field and press Enter.
4. Set the **Service Module Name** field to `/usr/lib/security/pki/JSML.sml`. This is the default SML load module. This field maps to the **program** attribute in the **/usr/lib/security/pki/ca.cfg** file.

5. Ignore the **Pathname of CA's Certificate** field. This field maps to the **certfile** attribute in the **/usr/lib/security/pki/ca.cfg** file.
6. Set the **Pathname of CA's Trusted Key** field to a URI that is the location of the trusted keystore on the local system. Only file-based keystores are supported. The typical location for the trusted keystore is in the **/usr/lib/security/pki** directory. (See "Configuring the Certificate Authentication Service client" on page 106.) This field maps to the **trustedkey** attribute in the **/usr/lib/security/pki/ca.cfg** file.
7. Set the **URI of the Certificate Authority Server** field to a URI that is the location of the CA (`cmp://myserver:1077`). This field maps to the **server** attribute in the **/usr/lib/security/pki/ca.cfg** file.
8. Ignore the **Certificate Distribution Point** field. This field maps to the **cdp** attribute in the **/usr/lib/security/pki/ca.cfg** file.
9. Set the **Certificate Revocation List (CRL) URI** field. This field specifies the URI that should be set to the location of the certificate revocation list for this CA. This is typically an LDAP URI, for example:

   ```
   ldap://crlserver/o=XYZ,c=us
   ```

   This field maps to the **crl** attribute in the **/usr/lib/security/pki/ca.cfg** file.
10. The **Default Certificate Distinguished Name** field specifies the baseline DN used when creating certificates (for example, `o=XYZ,c=us`). This field is *not* required. This field maps to the **dn** attribute in the **/usr/lib/security/pki/ca.cfg** file.
11. The **Default Certificate Subject Alternate Name URI** field specifies the default subject alternate name URI used when creating certificates if a subject alternate name URI is not provided at creation time. This field is *not* required. This field maps to the **url** attribute in the **/usr/lib/security/pki/ca.cfg** file.
12. The **Public Key Algorithm** field specifies the public key algorithm used when creating a certificate. The choices are `RSA` and `DSA`. If neither are specified, the system defaults to `RSA`. This field maps to the **algorithm** attribute in the **/usr/lib/security/pki/ca.cfg** file.
13. The **Public Key Size (in bits)** field specifies the bit size of the public key algorithm. This field is in bits, not bytes, and this value may be rounded up by the underlying public key mechanism to support the next feasible byte size. (Typically, rounding occurs when the number of bits is not a even multiple of 8). Example values are `512`, `1024`, and `2048`. If this field is not specified, the system defaults to `1024` bits. This field maps to the **keysize** attribute in the **/usr/lib/security/pki/ca.cfg** file.
14. The **MAX. Communications Retries** field specifies the number of times the system attempts to contact the CA (when creating or revoking a certificate) before giving up. The system defaults to 5 attempts. This field maps to the **retries** attribute in the **/usr/lib/security/pki/ca.cfg** file.
15. The **Signing Hash Algorithm** field specifies the hash algorithm used when signing an authentication certificate. The choices are `MD2`, `MD5`, and `SHA1`. The system defaults to `MD5`. This field maps to the **signinghash** attribute in the **/usr/lib/security/pki/ca.cfg** file.
16. Press Enter to commit the changes.

Change/Show Certificate Authority accounts:

This section provides the steps to change/show the Certificate Authority (CA) accounts.

1. Run PKI SMIT, as follows:

   ```
   smitty pki
   ```

2. Select **Change/Show CA Accounts**.
3. Type `local` for the **Certificate Authority Name** field and press Enter.
4. The **Certificate Creation Reference Number** field specifies the CA's reference number used in creating a certificate. The creation reference number must be composed of all digits and be at least 7 digits in length. The reference number is defined by the CA. (See "Creating the Certificate Authority" on page 105.) This field maps to the **carefnum** attribute in the **/usr/lib/security/pki/acct.cfg** file.
5. The **Certificate Creation Password** field specifies the CA's reference password used when creating a certificate. The creation password must be composed of 7-bit ASCII alpha-numerics and be at least 12 characters in length. The creation password is defined at the CA and must be the matching

password to the creation reference number above. (See “Creating the Certificate Authority” on page 105.) This field maps to the **capasswd** attribute in the **/usr/lib/security/pki/acct.cfg** file.

6. The **Certificate Revocation Reference Number** field specifies the reference number used when revoking a certificate. The revocation reference number must be composed of all digits and be at least 7 digits in length. The revocation reference number is sent to the CA during each certificate creation and is associated with the certificate by the CA. To revoke a certificate, the same revocation reference number (and revocation password) must be sent during revocation as was sent when creating the certificate. This field maps to the **rvrefnum** attribute in the **/usr/lib/security/pki/acct.cfg** file.
7. The `Certificate Revocation Password` field specifies the reference password used when revoking a certificate. The revocation password must be composed of 7-bit ASCII alpha-numerics and be at least 12 characters in length. The revocation password is sent to the CA during each certificate creation and is associated with the certificate by the CA. To revoke a certificate, the same revocation password (and revocation reference number) must be sent during revocation as was sent when creating the certificate. This field maps to the **rvpasswd** attribute in the **/usr/lib/security/pki/acct.cfg** file.
8. The **Trusted Key Label** field specifies the label (sometimes called *alias*) of the trusted signing key located in the trusted keystore. The trusted key label value is the value from “Creating the trusted signing key” on page 105. This field maps to the **keylabel** attribute in the **/usr/lib/security/pki/acct.cfg** file.
9. The **Trusted Key Password** field specifies the password of the trusted signing key located in the trusted keystore. The trusted key password value is the value from “Creating the trusted signing key” on page 105. This field maps to the **keypasswd** attribute in the **/usr/lib/security/pki/acct.cfg** file.
10. Press Enter to commit the changes.

Adding a Certificate Authority LDAP Account:

This section provides the steps to add a Certificate Authority (CA) LDAP account.

1. Run PKI SMIT, as follows
   ```
   smitty pki
   ```
2. Select **Add an LDAP Account**.
3. The **Administrative User Name** field specifies the LDAP administrative account DN. The administrative user name for the CA LDAP account is the same name used in both “Configuring the LDAP server” on page 101 and “Configuring the Certificate Authentication Service server to work with LDAP” on page 104. The value should be `cn=admin`. It is used by the client side to communicate with the LDAP server when accessing CA LDAP data. This field maps to the **ldappkiadmin** attribute in the **/usr/lib/security/pki/acct.cfg** file. For example:
   ```
   ldappkiadmin = "cn=admin"
   ```
4. The **Administrative Password** field specifies the LDAP administrative account password. The administrative password is the same password used in both “Configuring the LDAP server” on page 101 and “Configuring the Certificate Authentication Service server to work with LDAP” on page 104. This field maps to the **ldappkiadmpwd** attribute in the **/usr/lib/security/pki/acct.cfg** file. For example:
   ```
   ldappkiadmpwd = secret
   ```
5. The **Server Name** field specifies the name of the LDAP server and must be defined in every LDAP stanza. The value is a single LDAP server name. This field maps to the **ldapservers** attribute in the **/usr/lib/security/pki/acct.cfg** file. For example:
   ```
   ldapservers = ldapserver.mydomain.com
   ```
6. The **Suffix** field specifies the DN suffix for the directory information tree where the data resides. The suffix is the value of the **ibm-slapdSuffix** attribute used in “Configuring the Certificate Authentication Service server to work with LDAP” on page 104. This attribute must be defined in every LDAP stanza. This field maps to the **ldapsuffix** attribute in the **/usr/lib/security/pki/acct.cfg** file. For example:
   ```
   ldapsuffix = "ou=aix,cn=us"
   ```

7. Press Enter to commit the changes.

Add a PKI Per-User LDAP account:

Use this procedure to add a PKI Per-User LDAP account.

Perform the same steps as in “Adding a Certificate Authority LDAP Account” on page 108, except use the values used in the **Adding the PKI Suffix** and **ACL Database Entries** step in “Configuring the LDAP Server for PKI” on page 102. Use the following values:

- Administrative User Name (`ou=pkidata,cn=aixdata`),
- Administrative Password (*`password`*),
- Server Name (*`site specific`*),
- Suffix (`ou=pkidata,cn=aixdata`).

Press Enter to commit the changes.

Change/Show the Policy:

This section provides the steps to change/show the policy.

1. Run PKI SMIT, as follows:

   ```
   smitty pki
   ```

2. Select **Change / Show the Policy**.

- The **Create Certificates for New Users** field specifies whether the **mkuser** command generates a certificate and keystore for the new user (*new*), or if the administrator provides a certificate and keystore after the user is created (**get**). This field maps to the **cert** attribute of the newuser stanza in the **/usr/lib/security/pki/policy.cfg** file.
- The **Certificate Authority Name** field specifies the CA used by the **mkuser** command when generating a certificate. The field value must be a stanza name found in the **ca.cfg** file; for example, *local*. This field maps to the **ca** attribute of the newuser stanza in the **/usr/lib/security/pki/policy.cfg** file.
- The **Initial User Password** field specifies the password used by the **mkuser** command when creating a user’s keystore. This field maps to the **passwd** attribute of the newuser stanza in the **/usr/lib/security/pki/policy.cfg** file.
- The **Certificate Version** field specifies the certificate version used by the **mkuser** command when generating a certificate. Currently, the only supported value is 3, which represents X.509v3. This field maps to the **version** attribute of the newuser stanza in the **/usr/lib/security/pki/policy.cfg** file.
- The **Public Key Size** field specifies the size (in bits) of the public key used by the **mkuser** command when generating a certificate. This field maps to the **keysize** attribute of the newuser stanza in the **/usr/lib/security/pki/policy.cfg** file.
- The **Keystore Location** field specifies the keystore directory in URI format used by the **mkuser** command when creating a keystore. This field maps to the **keystore** attribute of the newuser stanza in the **/usr/lib/security/pki/policy.cfg** file.
- The **Validity Period** field specifies the certificate’s requested validity period used by the **mkuser** command when generating a certificate. The requested validity period may or may not be honored by the CA when creating the certificate. The period can be specified in seconds, days, or years. If just a number is provided, it is assumed to be in seconds. If the letter `d` immediately follows the number, it is interpreted as days. If the letter `y` immediate follows the number, it is interpreted as years. Example values are:
  - 1y (for 1 year)
  - 30d (for 30 days)
  - 2592000 (for 30 days represented in seconds)

  This field maps to the **validity** attribute of the newuser stanza in the **/usr/lib/security/pki/policy.cfg** file.

- The **Replicate Non-Local Certificates** field specifies whether the **certlink** command saves a copy of a certificate (Yes) or just the link to the certificate (No). This field maps to the **replicate** attribute of the storage stanza in the **/usr/lib/security/pki/policy.cfg** file.
- The **Check Certificate Revocation Lists** field specifies whether the **certadd** and **certlink** commands check the CRL before performing their tasks (Yes) or not (No). This field maps to the **check** attribute of the crl stanza in the **/usr/lib/security/pki/policy.cfg** file.
- The **Default Communications Timeout** field specifies the timeout period in seconds used by the **certadd** and **certlink** commands when requesting certificate information using HTTP (for example, retrieving CRLs). This field maps to the **timeout** attribute of the comm stanza in the **/usr/lib/security/pki/policy.cfg** file.

methods.cfg file:

The **methods.cfg** file specifies the definitions of the authentication grammar used by the **registry** and **SYSTEM** attributes. Specifically, this is where the authentication grammar for **PKILDAP** (for PKI using LDAP) and **FPKI** (*files* PKI) must be defined and added by the system administrator.

Below is a typical **methods.cfg** definition. The stanza names **PKI**, **LDAP**, and **PKILDAP** are arbitrary names and can be changed by the administrator. This section uses these stanza names throughout for consistency.

```
PKI:
  program = /usr/lib/security/PKI
  options = authonly

LDAP:
  program = /usr/lib/security/LDAP

PKILDAP:
  options = auth=PKI,db=LDAP
```

To support roaming users, use the same **methods.cfg** file stanza names and attribute values across all systems that support roaming users.

Administration configuration examples

The following examples show typical administration configuration tasks.

Creating a new PKI user account

To create a new PKI user account, use the **mkuser** command and the appropriate **/usr/lib/security/methods.cfg** stanza name (PKILDAP). Depending on the attribute settings in the **/usr/lib/security/pki/policy.cfg** file, the **mkuser** command can automatically create a certificate for the user. Below is a **mkuser** command example that creates the user account bob:

```
mkuser -R PKILDAP SYSTEM="PKILDAP" registry=PKILDAP bob
```

Converting a non-PKI user account to a PKI user account

There are two different approaches for converting a non-PKI user account into a PKI user account. The first approach allows the system administrator access to the user's private keystore initially, which might be acceptable in a given environment, but is the quickest way to convert a user. The second way requires interaction between the user and system administrator, which might take more time to setup.

Both examples use the following assumptions:

- **cas.server** and **cas.client** are already installed, configured, and working.
- **PKILDAP** is defined in **methods.cfg** as shown in "methods.cfg file."

Example 1:

With root authority, the system administrator can perform the following commands for user account bob:

```
certcreate -f cert1.der -l auth_lbl1 cn=bob bob  # Create & save cert in cert1.der.
certadd -f cert1.der -l auth_lbl1 auth_tag1 bob  # Add cert to LDAP as auth_tag1.
certverify auth_tag1 bob                         # Verify & sign the cert in LDAP.
chuser SYSTEM="PKILDAP" registry=PKILDAP bob     # Change account type to PKILDAP.
chuser -R PKILDAP auth_cert=auth_tag1 bob        # Set the user's auth certificate.
```

Then, have user bob change his password on the keystore using the **keypasswd** command.

Example 2:

Have user bob run the first 3 commands of example 1 above (**certcreate**, **certadd**, **certverify**), creating his own certificate and keystore. Then have the system administrator perform the last two **chuser** commands of example 1 above.

Creating and adding an authentication certificate

If a PKI user requires a new authentication certificate, the user can create a new certificate and have the system administrator make it the user's authentication certificate. Below is an example of user bob creating a certificate and the system administrator making the certificate the authentication certificate.

```
# Logged in as user account bob:
certcreate -f cert1.der -l auth_lbl1 cn=bob  # Create & save cert in cert1.der.
certadd -f cert1.der -l auth_lbl1 auth_tag1  # Add cert to LDAP as auth_tag1.
certverify auth_tag1                         # Verify & sign the cert in LDAP.
# As the system adminstrator:
chuser -R PKILDAP auth_cert=auth_tag1 bob    # Set the user's auth certificate.
```

Changing the default new-keystore password

Edit the **passwd** attribute value of the newuser stanza in the **/usr/lib/security/pki/policy.cfg** file to modify the password used to create the keystores of new PKI users.

Handling a compromised trusted signing key

The file that contains the trusted signing key needs to be replaced and the user authentication certificates need to be re-signed.

Handling a compromised user private key

If a user's private key is compromised, the user or the administrator should revoke the certificate using the appropriate reason code, other users that use the public key should be notified of the compromise and, depending on the purpose of the private/public key, a new certificate should be issued. If the certificate was used as the user's authentication certificate, then another certificate (either the new certificate or an existing non-promised certificate owned by the user) should be added as the new authentication certificate.

Handling a compromised keystore or keystore password

Change the password on the keystore. Revoke all the user's certificates. Create new certificates for the user including a new authentication certificate. The compromised private keys may still be useful to the user for accessing previously encrypted data.

Moving a user's keystore or changing the name of a user's keystore

Every user certificate maintained in LDAP contains the keystore location of its matching private key. To move a user's keystore from one directory to another or to change the name of the keystore, requires changing the LDAP keystore location and name associated with the user's certificates. If the user uses multiple keystores, then extra care must be taken to change only the LDAP information of the certificates affected by the keystore change.

To move a keystore from **/var/pki/security/keys/user1.p12** to **/var/pki/security1/keys/user1.p12**:

```
# As root...

cp /var/pki/security/keys/user1.p12 /var/pki/security1/keys/user1.p12

# Retrieve a list of all the certificates associated with the user.
certlist ALL user1

# For each certificate associated with the keystore, do the following:
# A) Retrieve the certificate's private key label and its "verified" status.
# B) Retrieve the certificate from LDAP.
# C) Replace the certificate in LDAP using the same private key label,
# but the new keystore path name.
# D) If the certificate was previously verified, it must verified again.
# (Step D requires the password to the keystore.)

# Example modifying one certificate.
# Assume:

# username: user1

# cert tag: tag1

# key label: label1

# Retrieve the certificate's private key label.
certlist -a label tag1 user1

# Retrieve the certificate from LDAP and place it in file cert.der.
certget -f cert.der tag1 user1

# Replace the certificate in LDAP.
certadd -r -f cert.der -p /var/pki/security1/keys/user1.p12 -l label1 tag1 user1

# Re-verify the certificate if it was previously verified.
# (Need to know the keystore password.)
certverify tag1 user1
```

Pluggable Authentication Modules

The pluggable authentication module (PAM) framework provides system administrators with the ability to incorporate multiple authentication mechanisms into an existing system through the use of pluggable modules.

Applications enabled to make use of PAM can be *plugged-in* to new technologies without modifying the existing applications. This flexibility allows administrators to do the following:

- Select any authentication service on the system for an application
- Use multiple authentication mechanisms for a given service
- Add new authentication service modules without modifying existing applications
- Use a previously entered password for authentication with multiple modules

The PAM framework consists of a library, pluggable modules, and a configuration file. The PAM library implements the PAM application programming interface (API) and serves to manage PAM transactions and invoke the PAM service programming interface (SPI) defined in the pluggable modules. Pluggable modules are dynamically loaded by the library based on the invoking service and its entry in the configuration file. Success is determined not only by the pluggable module but also by the behavior defined for the service.

Through the concept of *stacking*, a service can be configured to authenticate through multiple authentication methods. If supported, modules can also be configured to use a previously submitted password rather than prompting for additional input.

The system administrator can configure an AIX system to use PAM through modification of the **auth_type** attribute in the usw stanza of the**/etc/security/login.cfg** file. Setting `auth_type = PAM_AUTH` will configure PAM enabled commands to invoke the PAM API directly for authentication rather than use the historic AIX authentication routines. This configuration is a run-time decision and does not require a reboot of the system to take affect. For further information about the **auth_type** attribute, see the **/etc/security/login.cfg** file reference. The following native AIX commands and applications have been modified to recognize the **auth_type** attribute and thus been enabled for PAM authentication:

- login
- passwd
- su
- ftp
- telnet
- rlogin
- rexec
- rsh
- snappd
- imapd

The following illustration shows the interaction between PAM-enabled applications, PAM library, configuration file, and PAM modules on a system that has been configured to use PAM. PAM enabled applications invoke the PAM API in the PAM library. The library determines the appropriate module to load based on the application entry in the configuration file and calls the PAM SPI in the module. Communication occurs between the PAM module and the application through the use of a conversation function implemented in the application. Success or failure from the module and the behavior defined in the configuration file then determine if another module needs to be loaded. If so, the process continues; otherwise, the result is passed back to the application.

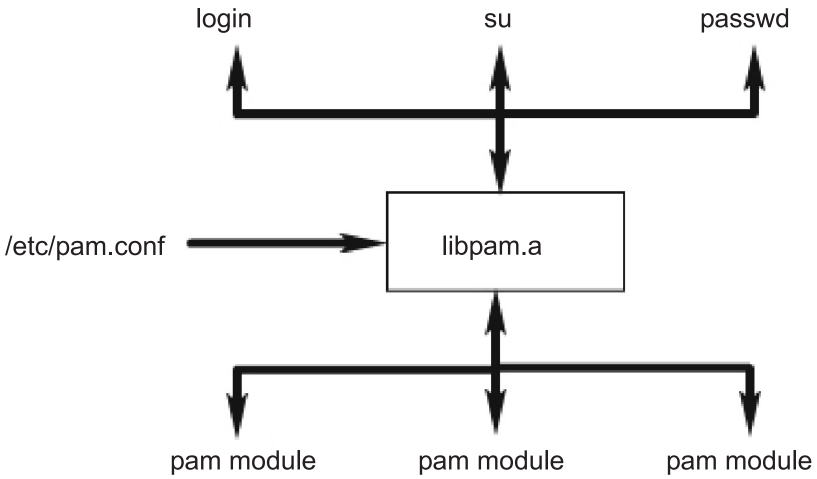


Figure 3. PAM Framework and Entities. This illustration shows how PAM enabled commands use the PAM library to access the appropriate PAM module.

PAM library

The PAM **library,/usr/lib/libpam.a**, contains the PAM API that serves as a common interface to all PAM applications and also controls module loading.

Modules are loaded by the PAM library based on the stacking behavior defined in the **/etc/pam.conf** file.

The following PAM API functions invoke the corresponding PAM SPI provided by a PAM module. For example, the pam_authenticate API invokes the pam_sm_authenticate SPI in a PAM module.

- `pam_sm_authenticate`
- `pam_sm_setcred`
- `pam_acct_mgmt`
- `pam_open_session`
- `pam_close_session`
- `pam_chauthtok`

Also provided in the PAM library are several functions that enable an application to invoke PAM modules and also to pass information to PAM modules. The following PAM framework APIs are implemented in AIX:

pam_start	Establish a PAM session
pam_end	Terminate a PAM session
pam_get_data	Retrieve module-specific data
pam_set_data	Set module-specific data
pam_getenv	Retrieve the value of a defined PAM environment variable
pam_getenvlist	Retrieve a list of all of the defined PAM environment variables and their values
pam_putenv	Set a PAM environment variable
pam_get_item	Retrieve common PAM information
pam_set_item	Set common PAM information
pam_get_user	Retrieve user name
pam_strerror	Get PAM standard error message

PAM modules

PAM modules allow multiple authentication mechanisms to be used collectively or independently on a system.

A given PAM module must implement at least one of four module types. The module types are described as follows, along with the corresponding PAM SPIs that are required to conform to the module type.

Authentication Modules
Authenticate users and set, refresh, or destroy credentials. These modules identify user based on their authentication and credentials.

Authentication module functions:

- pam_sm_authenticate
- pam_sm_setcred

Account Management Modules
Determine validity of the user account and subsequent access after identification from authentication module. Checks performed by these modules typically include account expiration and password restrictions.

Account management module function:

- pam_sm_acct_mgmt

Session Management Modules
Initiate and terminate user sessions. Additionally, support for session auditing may be provided.

Session management module functions:

- pam_sm_open_session

- pam_sm_close_session

Password Management Modules

Perform password modification and related attribute management.

Password management module functions:

- pam_sm_chauthtok

PAM configuration file

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
```

Where:

service_name
Specifies the name of the service. The keyword OTHER is used to define the default module to use for applications not specified in an entry.

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.

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, then the PAM library will prepend to the module name either `/usr/lib/security` for 32-bit services or `/usr/lib/security/64` for 64-bit services.

module_options
Specifies a space-delimited list of options that can be passed to the service modules. 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 incorrect 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 because this denotes a comment.

PAM supports a concept typically referred to as ″stacking″, allowing 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, with the final result determined by the **control_flag** field specified for each entry. Valid values for the **control_flag** field and the corresponding behavior in the stack are as follows:

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.

The following **/etc/pam.conf** subset is an example of stacking in the **auth** module type for the login service.

```
#
# PAM configuration file /etc/pam.conf
#

# 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 of configuration file contains three entries for the login service. Having specified both **pam_ckfile** and **pam_aix** as required, both modules will be run and both must be successful for the overall result to be successful. 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.

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.

Adding a PAM module

This section provides the steps to add a PAM module.

1. Place the 32-bit version of the module in the **/usr/lib/security** directory and the 64-bit version of the module in **/usr/lib/security/64** directory.
2. Set file ownership to `root` and permissions to `555`. The PAM library does not load any module not owned by the root user.
3. Update the **/etc/pam.conf** configuration file to include the module in entries for the desired service names.
4. Test the affected services to ensure their functionality. Do not log off the system until a login test has been performed.

Changing the /etc/pam.conf file

There are a few thing you should consider before changing the /etc/pam.conf file.

When changing the **/etc/pam.conf** configuration file, consider the following requirements:

- The file should always be owned by the root user and group security. Permission on the file needs to be 644 to allow everyone read access but only allow root to modify.
- 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 determine which control flags and options 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.

Enabling PAM debug

The PAM library can provide debug information during execution. After enabling the system to collect debug output, the information gathered can be used to track PAM-API invocations and determine failure points in the current PAM setup.

To enable PAM debug output, perform these steps:

1. Create an empty file at **/etc/pam_debug**. The PAM library checks for existence of the **/etc/pam_debug** file and if found, enables syslog output.
2. Edit the **/etc/syslog.conf** file to contain the appropriate entries for the desired levels of messages.
3. Restart the **syslogd** daemon so that configuration changes are recognized.
4. When the PAM application is restarted, debug messages will be collected in the output file defined in the **/etc/syslog.conf** configuration file.

pam_aix module

The pam_aix module is a PAM module that provides PAM-enabled applications access to AIX security services by providing interfaces that call the equivalent AIX services where they exist.

These services are in turn performed by a loadable authentication module or the AIX built-in function based on the user's definition and the corresponding setup in the **methods.cfg** file. Any error codes generated during execution of an AIX service are mapped to the corresponding PAM error code.

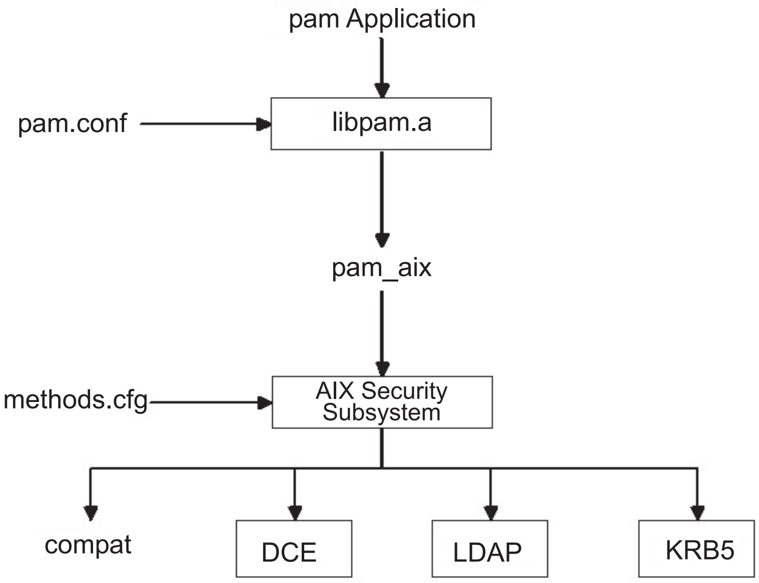


Figure 4. PAM Application to AIX Security Subsystem Path

This illustration shows the path that a PAM application API call will follow if the **/etc/pam.conf** file is configured to make use of the **pam_aix** module. As shown in the diagram, the integration allows users to be authenticated by any of the loadable authentication modules (DCE, LDAP, or KRB5) or in AIX files (compat).

The **pam_aix** module is installed in the **/usr/lib/security** directory. Integration of the **pam_aix** module requires that the **/etc/pam.conf** file be configured to make use of the module. Stacking is still available but is not shown in the following example of the **/etc/pam.conf** file:

```
#
# Authentication management
#
OTHER   auth     required       /usr/lib/security/pam_aix
```

```
#
# Account management
#
OTHER   account  required       /usr/lib/security/pam_aix

#
# Session management
#
OTHER   session  required       /usr/lib/security/pam_aix

#
# Password management
#
OTHER   password required       /usr/lib/security/pam_aix
```

The **pam_aix** module has implementations for the pam_sm_authenticate, pam_sm_chauthok and pam_sm_acct_mgmt SPI functions. The pam_sm_setcred, pam_sm_open_session, and pam_sm_close_session SPI are also implemented in the pam_aix module, but these SPI functions return PAM_SUCCESS invocations.

The following is an approximate mapping of PAM SPI calls to the AIX security subsystem:

```
 PAM SPI                   AIX
=========                 =====
pam_sm_authenticate   -->  authenticate
pam_sm_chauthtok      -->  passwdexpired, chpass
                           Note: passwdexpired is only checked if the
                           PAM_CHANGE_EXPIRED_AUTHTOK flag is passed in.
pam_sm_acct_mgmt      -->  loginrestrictions, passwdexpired
pam_sm_setcred        -->  No comparable mapping exists, PAM_SUCCESS returned
pam_sm_open_session   -->  No comparable mapping exists, PAM_SUCCESS returned
pam_sm_close_session  -->  No comparable mapping exists, PAM_SUCCESS returned
```

Data intended to be passed to the AIX security subsystem can be set using either the pam_set_item function prior to module use, or the pam_aix module for data if it does not already exist.

PAM loadable authentication module

AIX security services can be configured to call PAM modules through the use of the existing AIX loadable authentication module framework.

Note: Prior to AIX 5.3 a loadable authentication module PAM was used to provide PAM authentication to native AIX applications. Due to differences in behavior between this solution and a true PAM solution, the PAM loadable authentication module is no longer the recommended means to provide PAM authentication to native AIX applications. Instead, the auth_type attribute in the **usw** stanza of **/etc/security/login.cfg** should be set to PAM_AUTH to enable PAM authentication in AIX. For more information on the auth_type attribute, see **/etc/security/login.cfg**. Use of the PAM loadable authentication module is still supported, but it is deprecated. You should use the auth_type attribute to enable PAM authentication.

When the /usr/lib/security/methods.cfg file is set up correctly, the PAM load module routes AIX security services (passwd, login, and so on) to the PAM library. The PAM library checks the **/etc/pam.conf** file to determine which PAM module to use and then makes the corresponding PAM SPI call. Return values from PAM are mapped to AIX error codes and returned to the calling program.

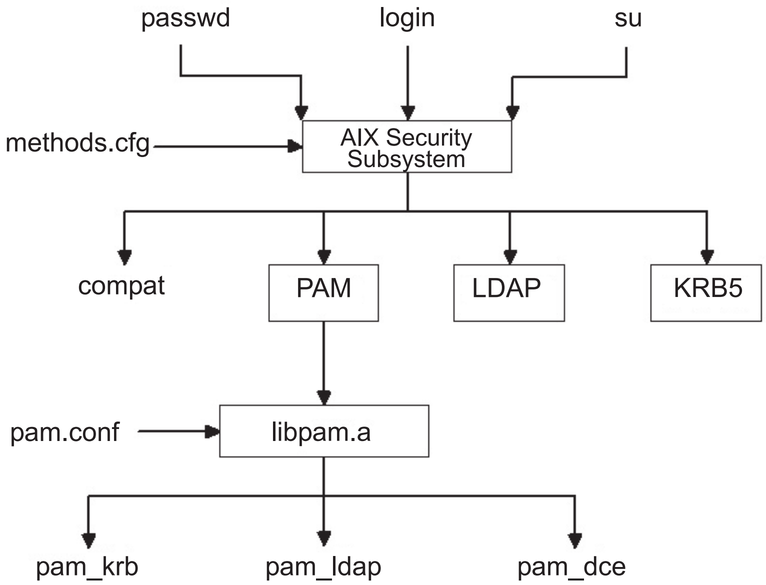


Figure 5. AIX Security Service to PAM Module Path

This illustration shows the path that an AIX security service call takes when PAM is configured correctly. The PAM modules shown (pam_krb, pam_ldap, and pam_dce) are listed as examples of third-party solutions.

The PAM load module is installed in the **/usr/lib/security** directory and is an authentication-only module. The PAM module must be combined with a database to form a compound load module. The following example shows the stanzas that could be added to the **methods.cfg** file to form a compound PAM module with a database called files. The BUILTIN keyword for the **db** attribute designates the database as UNIX files.

```
PAM:
        program = /usr/lib/security/PAM

PAMfiles:
        options = auth=PAM,db=BUILTIN
```

Creating and modifying users is then performed by using the -R option with the administration commands and by setting the *SYSTEM* attribute when a user is created. For example:

```
mkuser -R PAMfiles SYSTEM=PAMfiles registry=PAMfiles pamuser
```

This action informs further calls to AIX security services (login, passwd, and so on) to use the PAM load module for authentication. While the **files** database was used for the compound module in this example, other databases, such as LDAP, can also be used if they are installed. Creating users as previously described will result in the following mapping of AIX security to PAM API calls:

```
         AIX                    PAM API
        =====                   =========
        authenticate       -->  pam_authenticate
        chpass             -->  pam_chauthtok
        passwdexpired      -->  pam_acct_mgmt
        passwdrestrictions -->  No comparable mapping exists, success returned
```

Customizing the **/etc/pam.conf** file allows the PAM API calls to be directed to the desired PAM module for authentication. To further refine the authentication mechanism, stacking can be implemented.

Data prompted for by an AIX security service is passed to PAM through the pam_set_item function because it is not possible to accommodate user dialog from PAM. PAM modules written for integration with the PAM module should retrieve all data with pam_get_item calls and should not attempt to prompt the user to input data because this is handled by the security service.

Loop detection is provided to catch possible configuration errors in which an AIX security service is routed to PAM and then a PAM module in turn attempts to call the AIX security service to perform the operation. Detection of this loop event will result in an immediate failure of the intended operation.

Note: The **/etc/pam.conf** file should *not* be written to make use of the pam_aix module when using PAM integration from an AIX security service to a PAM module because this will result in a loop condition.

OpenSSH software tools

OpenSSH software tools support the SSH1 and SSH2 protocols. The tools provide shell functions where network traffic is encrypted and authenticated.

OpenSSH is based on client and server architecture. OpenSSH runs the **sshd** daemon process on the AIX host and waits for the connection from clients. It supports public-key and private-key pairs for authentication and encryption of channels to ensure secure network connections and host-based authentication. For more information about OpenSSH, including the man pages, see the following Web site:

http://www.openssh.org

For more information about OpenSSH on AIX, see the following Web site, which has the latest **installp** format packages for AIX 5L:

`http://oss.software.ibm.com/developerworks/projects/opensshi`

This section explains how to install and configure OpenSSH on AIX.

The OpenSSH software is shipped on the AIX 5.3 Expansion Pack. This version of OpenSSH is compiled and packaged as installp packages using the openssh-3.8.p1 level of source code. The installp packages include the man pages and the translated message filesets. The OpenSSH program contained in the Expansion Pack CD-ROM media is licensed under the terms and conditions of the IBM International Program License Agreement (IPLA) for Non-Warranted Programs.

Before installing the OpenSSH installp format packages, you must install the Open Secure Sockets Layer (OpenSSL) software that contains the encrypted library. OpenSSL is available in RPM packages on the AIX Toolbox for Linux® Applications CD, or you can also download the packages from the following AIX Toolbox for Linux Applications Web site:

`http://www-1.ibm.com/servers/aix/products/aixos/linux/download.html`

Because the OpenSSL package contains cryptographic content, you must register on the Web site to download the packages. You can download the packages by completing the following steps:

1. Click the AIX Toolbox Cryptographic Content link on the right side of the AIX Toolbox for Linux Applications Web site.
2. Click I have not registered before.
3. Fill in the required fields in the form.
4. Read the license and then click **Accept License**. The browser automatically redirects to the download page.
5. Scroll down the list of cryptographic content packages until you see **openssl-0.9.6m-1.aix4.3.ppc.rpm** under OpenSSL — SSL Cryptographic Libraries.
6. Click the **Download Now!** button for the **openssl-0.9.6m-1.aix4.3.ppc.rpm**.

After you download the OpenSSL package, you can install OpenSSL and OpenSSH.

1. Install the OpenSSL RPM package using the **geninstall** command:

   ```
   # geninstall -d/dev/cd0 R:openssl-0.9.6m
   ```

   Output similar to the following displays:

   ```
   SUCCESSES
   ---------
   openssl-0.9.6m-3
   ```
2. Install the OpenSSH installp packages using the **geninstall** command:

   ```
   # geninstall -I"Y" -d/dev/cd0 I:openssh.base
   ```

   Use the **Y** flag to accept the OpenSSH license agreement after you have reviewed the license agreement.

   Output similar to the following displays:

   ```
   Installation Summary
   --------------------
   Name                        Level           Part        Event       Result
   -------------------------------------------------------------------------------
   openssh.base.client         3.8.0.5200      USR         APPLY       SUCCESS
   openssh.base.server         3.8.0.5200      USR         APPLY       SUCCESS
   openssh.base.client         3.8.0.5200      ROOT        APPLY       SUCCESS
   openssh.base.server         3.8.0.5200      ROOT        APPLY       SUCCESS
   ```

You can also use the SMIT **install_software** fast path to install OpenSSL and OpenSSH.

The following OpenSSH binary files are installed as a result of the preceding procedure:

scp File copy program similar to rcp

sftp Program similar to FTP that works over SSH1 and SSH2 protocol

sftp-server
SFTP server subsystem (started automatically by sshd daemon)

ssh Similar to the rlogin and rsh client programs

ssh-add
Tool that adds keys to ssh-agent

ssh-agent
An agent that can store private keys

ssh-keygen
Key generation tool

ssh-keyscan
Utility for gathering public host keys from a number of hosts

ssh-keysign
Utility for host-based authentication

ssh-rand-helper
A program used by OpenSSH to gather random numbers. It is used only on AIX 5.1 installations.

sshd Daemon that permits you to log in

The following general information covers OpenSSH:

- The **/etc/ssh** directory contains the **sshd** daemon and the configuration files for the **ssh** client command.
- The **/usr/openssh** directory contains the readme file and the original OpenSSH open-source license text file. This directory also contains the **ssh** protocol and Kerberos license text.

- The **sshd** daemon is under AIX SRC control. You can start, stop, and view the status of the daemon by issuing the following commands:

  ```
  startsrc -s sshd   OR startsrc -g ssh  (group)
  stopsrc -s sshd    OR stopsrc -g ssh
  lssrc -s sshd      OR lssrc -s ssh
  ```

  You can also start and stop the daemon by issuing the following commands:

  ```
  /etc/rc.d/rc2.d/Ksshd start
  ```

  OR

  ```
  /etc/rc.d/rc2.d/Ssshd start
  ```

  ```
  /etc/rc.d/rc2.d/Ksshd stop
  ```

  OR

  ```
  /etc/rc.d/rc2.d/Ssshd stop
  ```

- When the OpenSSH server fileset is installed, an entry is added to the **/etc/rc.d/rc2.d** directory. An entry is in inittab to execute run-level 2 processes (`l2:2:wait:/etc/rc.d/rc 2`), so the **sshd** daemon will start automatically at boot time. To prevent the daemon from starting at boot time, remove the **/etc/rc.d/rc2.d/Ksshd** and **/etc/rc.d/rc2.d/Ssshd** files.
- OpenSSH software logs information to SYSLOG.
- The IBM Redbook, *Managing AIX Server Farms*, provides information about configuring OpenSSH in AIX and is available at the following Web site:

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

- OpenSSH supports long user names (256 bytes), the same as the AIX base operating system. For more information on long user names, see the**mkuser** command.

OpenSSH images

Use the following steps to install the OpenSSH images:

1. Download the images from http://www-124.ibm.com/developerworks/projects/opensshi
2. Decompress the image package using the **uncompress** *packagename* command. For example:

   ```
   uncompress openssh361p2_52_nologin.tar.Z
   ```

3. Untar the package with the **tar -xvf** *packagename* command. For example:

   ```
    tar -xvf openssh361p2_52_nologin.tar
   ```

4. Run **inutoc**.
5. Run **smitty install**.
6. Select **Install and Update Software**.
7. Select **Update Installed Software to Latest Level (Update All)**.
8. Type a dot (**.**) in the field for **INPUT device / directory for software** and press Enter.
9. Scroll down to **ACCEPT new license agreements** and press the **Tab** key to change the field to **Yes**.
10. Press the Enter key twice to start the installation.

The OpenSSH images are base level images, not Program Temporary Fixes (PTFs). Upon installation, all of the previous version's code is overwritten with the new version's images.

Configuration of OpenSSH compilation

This section provides information about how the OpenSSH code is compiled for AIX.

When configuring OpenSSH for AIX 5.1, the output is similar to the following:

```
OpenSSH has been configured with the following options:
                   User binaries: /usr/bin
                 System binaries: /usr/sbin
             Configuration files: /etc/ssh
                 Askpass program: /usr/sbin/ssh-askpass
                    Manual pages: /usr/man
```

```
                          PID file: /etc/ssh
  Privilege separation chroot path: /var/empty
            sshd default user PATH: /usr/bin:/bin:/usr/sbin:/sbin:/usr/local/bin

                    Manpage format: man
                       PAM support: no
                KerberosIV support: no
                 KerberosV support: yes
                 Smartcard support: no
                       AFS support: no
                     S/KEY support: no
              TCP Wrappers support: no
              MD5 password support: no
       IP address in $DISPLAY hack: no
          Use IPv4 by default hack: no
           Translate v4 in v6 hack: no
                  BSD Auth support: no
              Random number source: ssh-rand-helper
     ssh-rand-helper collects from: Command hashing (timeout 200)

              Host: powerpc-ibm-aix5.1.0.0
          Compiler: cc
    Compiler flags: -O -D__STR31__
Preprocessor flags: -I. -I$(srcdir) -I/home/BUILD/test2debug/zlib-1.1.3/ -I/o
pt/freeware/src/packages/SOURCES/openssl-0.9.6m/include -I/usr/include -I/usr/in
clude/gssapi -I/usr/include/ibm_svc -I/usr/local/include $(PATHS) -DHAVE_CONFIG_
H
      Linker flags: -L. -Lopenbsd-compat/ -L/opt/freeware/lib/ -L/usr/local/lib
-L/usr/krb5/lib -blibpath:/opt/freeware/lib:/usr/lib:/lib:/usr/local/lib:/usr/kr
b5/lib
         Libraries:   -lz  -lcrypto -lkrb5 -lk5crypto -lcom_err

WARNING: you are using the builtin random number collection
service. Please read WARNING.RNG and request that your OS
vendor includes kernel-based random number collection in
future versions of your OS.
```

When configuring OpenSSH for AIX 5.2 the output is similar to the following:

```
OpenSSH has been configured with the following options:
                     User binaries: /usr/bin
                   System binaries: /usr/sbin
               Configuration files: /etc/ssh
                   Askpass program: /usr/sbin/ssh-askpass
                      Manual pages: /usr/man
                          PID file: /etc/ssh
  Privilege separation chroot path: /var/empty
            sshd default user PATH: /usr/bin:/bin:/usr/sbin:/sbin:/usr/local/bin

                    Manpage format: man
                       PAM support: no
                KerberosIV support: no
                 KerberosV support: yes
                 Smartcard support: no
                       AFS support: no
                     S/KEY support: no
              TCP Wrappers support: no
              MD5 password support: no
       IP address in $DISPLAY hack: no
          Use IPv4 by default hack: no
           Translate v4 in v6 hack: no
                  BSD Auth support: no
              Random number source: OpenSSL internal ONLY

              Host: powerpc-ibm-aix5.2.0.0
          Compiler: cc
    Compiler flags: -O -D__STR31__
    Preprocessor flags: -I/opt/freeware/src/packages/BUILD/openssl-0.9.6m/includ
e  -I/usr/local/include -I/usr/local/include
      Linker flags: -L/opt/freeware/src/packages/BUILD/openssl-0.9.6m  -L/usr/lo
cal/lib -L/usr/local/lib -blibpath:/usr/lib:/lib:/usr/local/lib:/usr/local/lib
         Libraries:   -lz  -lcrypto -lkrb5 -lk5crypto -lcom_err
```

When configuring OpenSSH for AIX 5.3 the output is similar to the following:

```
OpenSSH has been configured with the following options:
                     User binaries: /usr/bin
                   System binaries: /usr/sbin
               Configuration files: /etc/ssh
                   Askpass program: /usr/sbin/ssh-askpass
                      Manual pages: /usr/man
                          PID file: /etc/ssh
  Privilege separation chroot path: /var/empty
            sshd default user PATH: /usr/bin:/bin:/usr/sbin:/sbin:/usr/
                                    local/bin

                    Manpage format: man                       PAM support: no
                KerberosIV support: no
                 KerberosV support: yes
                 Smartcard support: no
                       AFS support: no
                     S/KEY support: no
              TCP Wrappers support: no
              MD5 password support: no
       IP address in $DISPLAY hack: no
          Use IPv4 by default hack: no
           Translate v4 in v6 hack: no
                  BSD Auth support: no
              Random number source: OpenSSL internal ONLY

              Host: powerpc-ibm-aix5.3.0.0
          Compiler: cc
    Compiler flags: -O -D__STR31__
    Preprocessor flags: -I/opt/freeware/src/packages/BUILD/openssl-0.9.6m/

    include  -I/usr/local/include -I/usr/local/include
      Linker flags: -L/opt/freeware/src/packages/BUILD/openssl-0.9.6m
      -L/usr/local/lib -L/usr/local/lib -blibpath:/usr/lib:/lib:/usr/local/
      lib:/usr/local/lib
         Libraries:   -lz  -lcrypto -lkrb5 -lk5crypto -lcom_err
```

OpenSSH and Kerberos Version 5 support

Kerberos is an authentication mechanism that provides a secure means of authentication for network users. It prevents transmission of clear text passwords over the network by encrypting authentication messages between clients and servers. In addition, Kerberos provides a system for authorization in the form of administering tokens, or credentials.

To authenticate a user using Kerberos, the user runs the **kinit** command to gain initial credentials from a central Kerberos server known as the KDC (Key Distribution Center). The KDC verifies the user and passes back to the user his initial credentials, known as a TGT (Ticket-Granting Ticket). The user can then start a remote login session using a service such as a Kerberized Telnet or OpenSSH, and Kerberos authenticates the user by gaining user credentials from the KDC. Kerberos performs this authentication without any need for user interaction, therefore users do not need to enter passwords to login. IBM's version of Kerberos is known as Network Authentication Service (NAS). NAS can be installed from the AIX Expansion Pack CDs. It is available in the **krb5.client.rte** and **krb5.server.rte** packages. Beginning in the July 2003 release of OpenSSH 3.6, OpenSSH supports Kerberos 5 authentication and authorization through NAS version 1.3.

OpenSSH version 3.8 and later supports Kerberos 5 authentication and authorization through NAS Version 1.4. Any migration from previous versions of NAS (Kerberos) needs to happen before updating OpenSSH. OpenSSH version 3.8.x will only work with NAS version 1.4 or later.

AIX has created OpenSSH with Kerberos authentication as an optional method. If the Kerberos libraries are not installed on the system, when OpenSSH runs Kerberos authentication is skipped and OpenSSH tries the next configured authentication method (such as AIX authentication).

After you install Kerberos, it is recommended that you read the Kerberos documentation before configuring the Kerberos servers. For more information about how to install and administer Kerberos, refer to the *IBM Network Authentication Service Version 1.3 for AIX : Administrator's and User's Guide* located in the **/usr/lpp/krb5/doc/html/***lang***/ADMINGD.htm** path.

Using OpenSSH with Kerberos

The following steps provide information on the initial setup that is required in order to use OpenSSH with Kerberos:

1. On your OpenSSH clients and servers, the **/etc/krb5.conf** file must exist. This file tells Kerberos which KDC to use, how long of a lifetime to give each ticket, and so on. The following is an example **krb5.conf** file:

   ```
   [libdefaults]
   ticket_lifetime = 600
   default_realm = OPENSSH.AUSTIN.XYZ.COM
   default_tkt_enctypes = des3-hmac-sha1 des-cbc-crc
   default_tgs_enctypes = des3-hmac-sha1 des-cbc-crc

   [realms]
   OPENSSH.AUSTIN.xyz.COM = {
       kdc = kerberos.austin.xyz.com:88
       kdc = kerberos-1.austin.xyz.com:88
       kdc = kerberos-2.austin.xyz.com:88
       admin_server = kerberos.austin.xyz.com:749
       default_domain = austin.xyz.com
   }

   [domain_realm]
       .austin.xyz.com = OPENSSH.AUSTIN.XYZ.COM
       kdc.austin.xyz.com = OPENSSH.AUSTIN.XYZ.COM
   ```

2. Also, you must add the following Kerberos services to each client machine's **/etc/services** file:

   ```
   kerberos      88/udp    kdc    # Kerberos V5 KDC
   kerberos      88/tcp    kdc    # Kerberos V5 KDC
   kerberos-adm  749/tcp          # Kerberos 5 admin/changepw
   kerberos-adm  749/udp          # Kerberos 5 admin/changepw
   krb5_prop     754/tcp          # Kerberos slave
                                  #  propagation
   ```

3. If your KDC is using LDAP as the registry to store user information, read "LDAP authentication load module" on page 69, and the Kerberos publications. Furthermore, make sure the following actions are performed:
   - KDC is running the LDAP client. You can start the LDAP client daemon with the **secldapclntd** command.
   - LDAP server is running the **slapd** LDAP server daemon.

4. On the OpenSSH server, edit the **/etc/ssh/sshd_config** file to contain the lines:

   ```
   KerberosAuthentication yes
   KerberosTicketCleanup yes
   GssapiAuthentication yes
   GssapiKeyExchange yes
   GssapiCleanupCreds yes
   UseDNS yes
   ```

   If UseDNS is set to **Yes**, the ssh server does a reverse host lookup to find the name of the connecting client. This is necessary when host-based authentication is used or when you want last login information to display host names rather than IP addresses.

   Note: Some ssh sessions stall when performing reverse name lookups because the DNS servers are unreachable. If this happens, you can skip the DNS lookups by setting UseDNS to `no`. If UseDNS is not explicitly set in the **/etc/ssh/sshd_config** file, the default value is `UseDNS yes`.

5. On the SSH server, run the **startsrc -g ssh** command to start the ssh server daemon.

6. On the SSH client machine, run the **kinit** command to gain initial credentials (a TGT). You can verify that you received a TGT by running the **klist** command. This shows all credentials belonging to you.
7. Connect to the server by running the **ssh** *username@servername* command.
8. If Kerberos is properly configured to authenticate the user, a prompt for a password will not display, and the user will be automatically logged into the SSH server.

Securing the network

These sections describe how to install and configure IP Security; how to identify necessary and unnecessary network services; auditing and monitoring network security.

TCP/IP security

If you installed the Transmission Control Protocol/Internet Protocol (TCP/IP) and Network File System (NFS) software, you can configure your system to communicate over a network.

This guide does not describe the basic concepts of TCP/IP, but rather describes security-related concerns of TCP/IP. For information on installing and the initial configuration of TCP/IP, refer to the Transmission Control Protocol/Internet Protocol section in *AIX 5L Version 5.3 System Management Guide: Communications and Networks*.

For any number of reasons, the person who administers your system might have to meet a certain level of security. For instance, the security level might be a matter of corporate policy. Or a system might need access to government systems and thus be required to communicate at a certain security level. These security standards might be applied to the network, the operating system, application software, even programs written by the person who administers your system.

This section describes the security features provided with TCP/IP, both in standard mode and as a secure system, and discusses some security considerations that are appropriate in a network environment.

After you install TCP/IP and NFS software, use the Web-based System Manager or the System Management Interface Tool (SMIT) **tcpip** fast path, to configure your system.

For more information on the **dacinet** command, refer to the *AIX 5L Version 5.3 Commands Reference*.

Operating system-specific security

Many of the security features, such as network access control and network auditing, available for TCP/IP are based on those available through the operating system.

The following sections outline TCP/IP security.

Network access control

The security policy for networking is an extension of the security policy for the operating system and consists of user authentication, connection authentication, and data security.

It consists of the following major components:

- User authentication is provided at the remote host by a user name and password in the same way as when a user logs in to the local system. Trusted TCP/IP commands, such as **ftp**, **rexec**, and **telnet**, have the same requirements and undergo the same verification process as trusted commands in the operating system.
- Connection authentication is provided to ensure that the remote host has the expected Internet Protocol (IP) address and name. This prevents a remote host from masquerading as another remote host.
- Data import and export security permits data at a specified security level to flow to and from network interface adapters at the same security and authority levels. For example, top-secret data can flow only between adapters that are set to the top-secret security level.

Network auditing

Network auditing is provided by TCP/IP, using the audit subsystem to audit both kernel network routines and application programs.

The purpose of auditing is to record those actions that affect the security of the system and the user responsible for those actions.

The following types of events are audited:

Kernel events

- Change configuration
- Change host ID
- Change route
- Connection
- Create socket
- Export object
- Import object

Application events

- Access the network
- Change configuration
- Change host ID
- Change static route
- Configure mail
- Connection
- Export data
- Import data
- Write mail to a file

Creation and deletion of objects are audited by the operating system. Application audit records suspend and resume auditing to avoid redundant auditing by the kernel.

Trusted path, trusted shell, and Secure Attention Key

The operating system provides the *trusted path* to prevent unauthorized programs from reading data from a user terminal. This path is used when a secure communication path with the system is required, such as when you are changing passwords or logging in to the system.

The operating system also provides the *trusted shell* (**tsh**), which executes only trusted programs that have been tested and verified as secure. TCP/IP supports both of these features, along with the *secure attention key* (SAK), which establishes the environment necessary for secure communication between you and the system. The local SAK is available whenever you are using TCP/IP. A remote SAK is available through the **telnet** command.

The local SAK has the same function in **telnet** that it has in other operating system application programs: it ends the **telnet** process and all other processes associated with the terminal in which **telnet** was running. Inside the telnet program, however, you can send a request for a trusted path to the remote system using the **telnet send sak** command (while in **telnet** command mode). You can also define a single key to initiate the SAK request using the **telnet set sak** command.

For more information about the Trusted Computing Base, see “Trusted Computing Base” on page 1.

TCP/IP command security

Some commands in TCP/IP provide a secure environment during operation. These commands are **ftp**, **rexec**, and **telnet**.

The **ftp** function provides security during file transfer. The **rexec** command provides a secure environment for running commands on a foreign host. The **telnet** function provides security for login to a foreign host.

The **ftp**, **rexec**, and **telnet** commands provide security during their operation only. That is, they do not set up a secure environment for use with other commands. For securing your system for other operations, use the **securetcpip** command. This command gives you the ability to secure your system by disabling the nontrusted daemons and applications, and by giving you the option of securing your IP layer network protocol as well.

The **ftp**, **rexec**, **securetcpip**, and **telnet** commands provide the following forms of system and data security:

ftp

The **ftp** command provides a secure environment for transferring files. When a user invokes the **ftp** command to a foreign host, the user is prompted for a login ID. A default login ID is shown: the user's current login ID on the local host. The user is prompted for a password for the remote host.

The automatic login process searches the local user's **$HOME/.netrc** file for the user's ID and password to use at the foreign host. For security, the permissions on the **$HOME/.netrc** file must be set to 600 (read and write by owner only). Otherwise, automatic login fails.

> **Note:** Because use of the **.netrc** file requires storage of passwords in a nonencrypted file, the automatic login feature of the **ftp** command is not available when your system has been configured with the **securetcpip** command. This feature can be reenabled by removing the **ftp** command from the tcpip stanza in the **/etc/security/config** file.

To use the file transfer function, the **ftp** command requires two TCP/IP connections, one for the File Transfer Protocol (FTP) and one for data transfer. The protocol connection is primary and is secure because it is established on reliable communicating ports. The secondary connection is needed for the actual transfer of data, and both the local and remote host verify that the other end of this connection is established with the same host as the primary connection. If the primary and secondary connections are not established with the same host, the **ftp** command first displays an error message stating that the data connection was not authenticated, and then it exits. This verification of the secondary connection prevents a third host from intercepting data intended for another host.

rexec

The **rexec** command provides a secure environment for executing commands on a foreign host. The user is prompted for both a login ID and a password.

An automatic login feature causes the **rexec** command to search the local user's **$HOME/.netrc** file for the user's ID and password on a foreign host. For security, the permissions on the **$HOME/.netrc** file must be set to 600 (read and write by owner only). Otherwise, automatic login fails.

> **Note:** Because use of the **.netrc** file requires storage of passwords in a nonencrypted file, the automatic login feature of **rexec** command is not available when your system is operating in secure. This feature can be reenabled by removing the entry from the tcpip stanza in the **/etc/security/config** file.

securetcpip

The **securetcpip** command enables TCP/IP security features. Access to commands that are not trusted is removed from the system when this command is issued. Each of the following commands is removed by running the **securetcpip** command:

- **rlogin** and **rlogind**
- **rcp**, **rsh**, and **rshd**
- **tftp** and **tftpd**
- **trpt**

The **securetcpip** command is used to convert a system from the standard level of security to a higher security level. After your system has been converted, you need not issue the **securetcpip** command again unless you reinstall TCP/IP.

telnet or **tn**

The **telnet** (TELNET) command provides a secure environment for login to a foreign host. The user is prompted for both a login ID and a password. The user's terminal is treated just like a terminal connected directly to the host. That is, access to the terminal is controlled by permission bits. Other users (group and other) do not have read access to the terminal, but they can write messages to it if the owner gives them write permission. The **telnet** command also provides access to a trusted shell on the remote system through the SAK. This key sequence differs from the sequence that invokes the local trusted path and can be defined within the **telnet** command.

Remote command execution access

Users on the hosts listed in the **/etc/hosts.equiv** file can run certain commands on your system without supplying a password.

The following table provides information about how to list, add, and remove remote hosts using Web-based System Manager, SMIT, or command line.

Table 9. Remote command execution access tasks

Task	SMIT fast path	Command or file	Web-based System Manager Management Environment
List Remote Hosts That Have Command Execution Access	**smit lshostsequiv**	view **/etc/hosts.equiv** file	**Software** → **Network** → **TCPIP (IPv4 and IPv6)** → **TCPIP Protocol Configuration** → **TCP/IP** → **Configure TCP/IP** → **Advanced Methods** → **Hosts File** → **Contents of /etc/hosts file** .
Add a Remote Host for Command Execution Access	**smit mkhostsequiv**	edit **/etc/hosts.equiv** file[note]	**Software** → **Network** → **TCPIP (IPv4 and IPv6)** → **TCPIP Protocol Configuration** → **TCP/IP** → **Configure TCP/IP** → **Advanced Methods** → **Hosts File** . In **Add/Change host entry**, complete the following fields: **IP Address**, **Host name**, **Alias(es)**, and **Comment**. Click **Add/Change Entry**, and click **OK**.
Remove a Remote Host from Command Execution Access	**smit rmhostsequiv**	edit **/etc/hosts.equiv** file[note]	**Software** → **Network** → **TCPIP (IPv4 and IPv6)** → **TCPIP Protocol Configuration** → **TCP/IP** → **Configure TCP/IP** → **Advanced Methods** → **Hosts File**. Select a host in **Contents of /etc/host file**. Click **Delete Entry** → **OK**.

Note: For more information about these file procedures, see the ″hosts.equiv File Format for TCP/IP″ in the *AIX 5L Version 5.3 Files Reference*.

Restricted file transfer program users

Users listed in the **/etc/ftpusers** file are protected from remote FTP access. For example, suppose that user A is logged into a remote system, and he knows the password of user B on your system. If user B is listed in the **/etc/ftpusers** file, user A cannot FTP files to or from user B's account, even though user A knows user B's password.

The following table provides information about how to list, add, and remove restricted users using Web-based System Manager, SMIT, or command line.

Remote FTP user tasks

Task	SMIT fast path	Command or file	Web-based System Manager Management Environment
List Restricted FTP Users	**smit lsftpusers**	view **/etc/ftpusers** file	**Software** → **Users** → **All Users**.
Add a Restricted User	**smit mkftpusers**	edit **/etc/ftpusers** file[Note]	**Software** → **Users** → **All Users** → **Selected** → **Add this User to Group**. Select a group, and click **OK**.
Remove a Restricted User	**smit rmftpusers**	edit **/etc/ftpusers** file[Note]	**Software** → **Users** → **All Users** → **Selected** → **Delete**.

Note: For more information about these file procedures, see the ″ftpusers File Format for TCP/IP″ in the *AIX 5L Version 5.3 Files Reference*.

Trusted processes

A trusted program, or trusted process, is a shell script, a daemon, or a program that meets a particular standard of security. These security standards are set and maintained by the U.S. Department of Defense, which also certifies some trusted programs.

Trusted programs are trusted at different levels. Security levels include A1, B1, B2, B3, C1, C2, and D, with level A1 providing the highest security level. Each security level must meet certain requirements. For example, the C2 level of security incorporates the following standards:

program integrity
Ensures that the process performs exactly as intended.

modularity
Process source code is separated into modules that cannot be directly affected or accessed by other modules.

principle of least privilege
States that at all times a user is operating at the lowest level of privilege authorized. That is, if a user has access only to view a certain file, then the user does not inadvertently also have access to alter that file.

limitation of object reuse
Keeps a user from, for example, accidentally finding a section of memory that has been flagged for overwriting but not yet cleared, and which might contain sensitive material.

TCP/IP contains several trusted daemons and many nontrusted daemons.

Examples of trusted daemons are as follows:

- **ftpd**
- **rexecd**
- **telnetd**

Examples of nontrusted daemons are as follows:

- **rshd**
- **rlogind**

- **tftpd**

For a system to be trusted, it must operate with a trusted computing base; that is, for a single host, the machine must be secure. For a network, all file servers, gateways, and other hosts must be secure.

Network Trusted Computing Base

The Network Trusted Computing Base (NTCB) consists of hardware and software for ensuring network security. This section defines the components of the NTCB as they relate to TCP/IP.

The hardware security features for the network are provided by the network adapters used with TCP/IP. These adapters control incoming data by receiving only data destined for the local system and broadcast data receivable by all systems.

The software component of the NTCB consists of only those programs that are considered as trusted. The programs and associated files that are part of a secure system are listed in the following tables on a directory-by-directory basis.

/etc directory

Name	Owner	Group	Mode	Permissions
gated.conf	root	system	0664	rw-rw-r—
gateways	root	system	0664	rw-rw-r—
hosts	root	system	0664	rw-rw-r—
hosts.equiv	root	system	0664	rw-rw-r—
inetd.conf	root	system	0644	rw-r—r—
named.conf	root	system	0644	rw-r—r—
named.data	root	system	0664	rw-rw-r—
networks	root	system	0664	rw-rw-r—
protocols	root	system	0644	rw-r—r—
rc.tcpip	root	system	0774	rwxrwxr—
resolv.conf	root	system	0644	rw-rw-r—
services	root	system	0644	rw-r—r—
3270.keys	root	system	0664	rw-rw-r—
3270keys.rt	root	system	0664	rw-rw-r—

/usr/bin directory

Name	Owner	Group	Mode	Permissions
host	root	system	4555	r-sr-xr-x
hostid	bin	bin	0555	r-xr-xr-x
hostname	bin	bin	0555	r-xr-xr-x
finger	root	system	0755	rwxr-xr-x
ftp	root	system	4555	r-sr-xr-x
netstat	root	bin	4555	r-sr-xr-x
rexec	root	bin	4555	r-sr-xr-x
ruptime	root	system	4555	r-sr-xr-x
rwho	root	system	4555	r-sr-xr-x
talk	bin	bin	0555	r-xr-xr-x

/usr/bin directory

Name	Owner	Group	Mode	Permissions
telnet	root	system	4555	r-sr-xr-x

/usr/sbin directory

Name	Owner	Group	Mode	Permissions
arp	root	system	4555	r-sr-xr-x
fingerd	root	system	0554	r-xr-xr—
ftpd	root	system	4554	r-sr-xr—
gated	root	system	4554	r-sr-xr—
ifconfig	bin	bin	0555	r-xr-xr-x
inetd	root	system	4554	r-sr-xr—
named	root	system	4554	r-sr-x—
ping	root	system	4555	r-sr-xr-x
rexecd	root	system	4554	r-sr-xr—
route	root	system	4554	r-sr-xr—
routed	root	system	0554	r-xr-x—-
rwhod	root	system	4554	r-sr-xr—
securetcpip	root	system	0554	r-xr-xr—
setclock	root	system	4555	r-sr-xr-x
syslogd	root	system	0554	r-xr-xr—
talkd	root	system	4554	r-sr-xr—
telnetd	root	system	4554	r-sr-xr—

/usr/ucb directory

Name	Owner	Group	Mode	Permissions
tn	root	system	4555	r-sr-xr-x

/var/spool/rwho directory

Name	Owner	Group	Mode	Permissions
rwho (directory)	root	system	0755	drwxr-xr-x

Data security and information protection

The security feature for TCP/IP does not encrypt user data transmitted through the network.

Identify any risk in communication that could result in the disclosure of passwords and other sensitive information, and based on that risk, apply appropriate countermeasures.

Using the TCP/IP security feature in a Department of Defense (DOD) environment might require adherence to DOD 5200.5 and NCSD-11 for communications security.

User based TCP port access control with discretionary access control for internet ports

Discretionary Access Control for Internet Ports (DACinet) features user-based access control for TCP ports for communication between AIX 5.2 hosts.

AIX 5.2 can use an additional TCP header to transport user and group information between systems. The DACinet feature allows the administrator on the destination system to control access based on the destination port, the originating user id and host.

Note: The DACinet facility is available only in CAPP/EAL4+ configured AIX systems.

In addition, the DACinet feature allows the administrator to restrict local ports for root only usage. UNIX systems like AIX treat ports below 1024 as privileged ports which can only be opened by root. AIX 5.2 allows you to specify additional ports above 1024 which can be opened only by root, therefore preventing users from running servers on well known ports.

Depending on the settings a non-DACinet system may or may not be able to connect to a DACinet system. Access is denied in the initial state of the DACinet feature. Once DACinet has been enabled, there is no way to disable DACinet.

The **dacinet** command accepts addresses which are specified as hostnames, dotted decimal host addresses, or network addresses followed by the length of the network prefix.

The following example specifies a single host which is known by the fully qualified host name *host.domain.org*:

```
host.domain.org
```

The following example specifies a single host which is known by the IP address 10.0.0.1:

```
10.0.0.1
```

The following example specifies the entire network which has the first 24 bits (the length of the network prefix) with a value of 10.0.0.0:

```
10.0.0.0/24
```

This network includes all IP addresses between 10.0.0.1 and 10.0.0.254.

Access control for TCP based services

DACinet uses the **/etc/rc.dacinet** startup file, and the configuration files it uses are **/etc/security/priv**, **/etc/security/services**, and **/etc/security/acl**.

Ports listed in **/etc/security/services** are considered exempt from the ACL checks. The file has the same format as **/etc/services**. The easiest way to initialize it is to copy the file from **/etc** to **/etc/security** and then delete all the ports for which ACLs should be applied. The ACLs are stored in two places. The currently active ACLs are stored in the kernel and can be read by running **dacinet aclls**. ACLs that will be reactivated at the next system boot by **/etc/rc.tcpip** are stored in **/etc/security/acl**. The following format is used:

```
service host/prefix-length [user|group]
```

Where the service can be specified either numerically or as listed in **/etc/services**, the host can be given either as a host name or a network address with a subnet mask specification and the user or group is specified with the u: or g: prefix. When no user or group is specified, then the ACL takes only the sending host into account. Prefixing the service with a - will disable access explicitly. ACLs are evaluated according to the first match. So you could specify access for a group of users, but explicitly deny it for a user in the group by placing the rule for this user in front of the group rule.

The **/etc/services** file includes two entries with port number values which are not supported in AIX 5.2. The system administrator must remove both lines from that file prior to executing the **mkCCadmin** command. Remove the following lines from the **/etc/services** file:

```
sco_printer     70000/tcp     sco_spooler    # For System V print IPC
sco_s5_port     70001/tcp     lpNet_s5_port  # For future use
```

DACinet usage examples:

For example, when using DACinet to restrict access to port TCP/25 inbound to root only with the DACinet feature, then only root users from other AIX 5.2 hosts can access this port, therefore limiting the possibilities of regular users to spoof e-mail by just telneting to port TCP/25 on the victim.

The following example shows how to configure the X protocol (X11) for root only access. Make sure that the X11 entry in **/etc/security/services** is removed, so that the ACLs will apply for this service.

Assuming a subnet of 10.1.1.0/24 for all the connected systems, the ACL entries to restrict access to the root user only for X (TCP/6000) in **/etc/security/acl** would be as follows:

```
      6000    10.1.1.0/24 u:root
```

When limiting Telnet service to users in the group friends, no matter from which system they are coming from, use the following ACL entry after having removed the telnet entry from **/etc/security/services**:

```
telnet    0.0.0.0/0   g:friends
```

Disallow user fred access to the web server, but allow everyone else access:

```
-80    0.0.0.0/0 u:fred
80     0.0.0.0/0
```

Privileged ports for running local services

To prevent regular users from running servers at specific ports, these ports can be designated as privileged.

Normally any user can open any port above 1024. For example, a user could place a server at port 8080, which is quite often used to run Web proxies or at 1080 where one typically finds a SOCKS server. The **dacinet setpriv** command can be used to add privileged ports to the running system. Ports that are to be designated as privileged when the system starts have to be listed in **/etc/security/priv**.

Ports can be listed in this file either with their symbolic name as defined in **/etc/services** or by specifying the port number. The following entries would disallow non-root users to run SOCKS servers or Lotus Notes® servers on their usual ports:

```
1080
lotusnote
```

Note: This feature does not prevent the user from running the programs. It will only prevent the user from running the services at the well known ports where those services are typically expected.

Network services

This section provides information about identifying and securing network services with open communication ports.

This section discusses:

- “Identifying network services with open communication ports” on page 136
- “Identifying TCP and UDP sockets” on page 138

Identifying network services with open communication ports

Client-server applications open communication ports on the server, allowing the applications to listen to incoming client requests.

Because open ports are vulnerable to potential security attacks, identify which applications have open ports and close those ports that are open unnecessarily. This practice is useful because it allows you to understand what systems are being made available to anyone who has access to the Internet.

To determine which ports are open, follow these steps:

1. Identify the services by using the **netstat** command as follows:

   ```
   # netstat -af inet
   ```

   The following is an example of this command output. The last column of the **netstat** command output indicates the state of each service. Services that are waiting for incoming connections are in the LISTEN state.

```
Active Internet connection (including servers)
Proto        Recv-Q          Send-Q  Local Address         Foreign Address   (state)
tcp4      0          0               *.echo                *.*               LISTEN
tcp4      0          0               *.discard             *.*               LISTEN
tcp4      0          0               *.daytime             *.*               LISTEN
tcp       0          0               *.chargen             *.*               LISTEN
tcp       0          0               *.ftp                 *.*               LISTEN
tcp4      0          0               *.telnet              *.*               LISTEN
tcp4      0          0               *.smtp                *.*               LISTEN
tcp4      0          0               *.time                *.*               LISTEN
tcp4      0          0               *.www                 *.*               LISTEN
tcp4      0          0               *.sunrpc              *.*               LISTEN
tcp       0          0               *.smux                *.*               LISTEN
tcp       0          0               *.exec                *.*               LISTEN
tcp       0          0               *.login               *.*               LISTEN
tcp4      0          0               *.shell               *.*               LISTEN
tcp4      0          0               *.klogin              *.*               LISTEN
udp4      0          0               *.kshell              *.*               LISTEN
udp4      0          0               *.echo                *.*
udp4      0          0               *.discard             *.*
udp4      0          0               *.daytime             *.*
udp4      0          0               *.chargen             *.*
udp4      0          0               *.time                *.*
udp4      0          0               *.bootpc              *.*
udp4      0          0               *.sunrpc              *.*
```

```
Active Internet connection (including servers)
Proto       Recv-Q             Send-Q  Local Address           Foreign Address   (state)
udp4     0        0                    255.255.255.255.ntp     *.*
udp4     0        0                    1.23.123.234.ntp        *.*
udp4     0        0                    localhost.domain.ntp    *.*
udp4     0        0                    name.domain..ntp        *.*
....................................
```

2. Open the **/etc/services** file and check the Internet Assigned Numbers Authority (IANA) services to map the service to port numbers within the operating system.

   The following is a sample fragment of the **/etc/services** file:

```
tcpmux                    1/tcp          # TCP Port Service Multiplexer
tcpmux                    1/tcp          # TCP Port Service Multiplexer
Compressnet               2/tcp          # Management Utility
Compressnet               2/udp          # Management Utility
Compressnet               3/tcp          # Compression Process
Compressnet               3/udp          Compression Process
Echo                      7/tcp
Echo                      7/udp
discard                   9/tcp          sink null
discard                   9/udp          sink null
..............
rfe                       5002/tcp       # Radio Free Ethernet
rfe                       5002/udp       # Radio Free Ethernet
rmonitor_secure           5145/tcp
rmonitor_secure           5145/udp
pad12sim                  5236/tcp
pad12sim                  5236/udp
sub-process               6111/tcp       # HP SoftBench Sub-Process Cntl.
sub-process               6111/udp       # HP SoftBench Sub-Process Cntl.
xdsxdm                    6558/ucp
xdsxdm                    6558/tcp
afs3-fileserver           7000/tcp       # File Server Itself
afs3-fileserver           7000/udp       # File Server Itself
af3-callback              7001/tcp       # Callbacks to Cache Managers
af3-callback              7001/udp       # Callbacks to Cache Managers
```

3. Close the unnecessary ports by removing the running services.

Note: Port 657 is used by Resource Monitoring and Control (RMC) for communication between nodes. You cannot block or otherwise restrict this port.

Identifying TCP and UDP sockets

Use the **lsof** command, a variant of the **netstat -af** command to identify TCP sockets that are in the LISTEN state and idle UDP sockets that are waiting for data to arrive.

For example, to display the TCP sockets in the LISTEN state and the UDP sockets in the IDLE state, run the **lsof** command as follows:

```
# lsof -i | egrep "COMMAND|LISTEN|UDP"
```

The output produced is similar to the following:

Command	PID	USER	FD	TYPE	DEVICE	SIZE/OFF	NODE	NAME
dtlogin	2122	root	5u	IPv4	0x70053c00	0t0	UDP	*:xdmcp
dtlogin	2122	root	6u	IPv4	0x70054adc	0t0	TCP	*:32768(LISTEN)
syslogd	2730	root	4u	IPv4	0x70053600	0t0	UDP	*:syslog
X	2880	root	6u	IPv4	0x70054adc	0t0	TCP	*:32768(LISTEN)
X	2880	root	8u	IPv4	0x700546dc	0t0	TCP	*:6000(LISTEN)
dtlogin	3882	root	6u	IPv4	0x70054adc	0t0	TCP	*:32768(LISTEN)
glbd	4154	root	4u	IPv4	0x7003f300	0t0	UDP	*:32803
glbd	4154	root	9u	IPv4	0x7003f700	0t0	UDP	*:32805
dtgreet	4656	root	6u	IPv4	0x70054adc	0t0	TCP	*:32768(LISTEN)

After identifying the process ID, you can obtain more information about the program by running the following command:

```
" # ps -fp PID#"
```

The output contains the path to the command name, which you can use to access the program's man page.

Internet Protocol security

IP Security enables secure communications over the Internet and within company networks by securing data traffic at the IP layer.

This allows individual users or organizations to secure traffic for all applications, without having to make any modifications to the applications. Therefore, the transmission of any data, such as e-mail or application-specific company data, can be made secure.

IP security overview

This section discusses the following topics:

- IP Security and the Operating System
- IP Security Features
- Security Associations
- Tunnels and Key Management

- Native Filtering Capability
- Digital Certificate Support
- Benefits of a Virtual Private Network

IP security and the operating system

The operating system uses IP Security (IPsec), which is an open, standard security technology developed by the Internet Engineering Task Force (IETF).

IPsec provides cryptography-based protection of all data at the IP layer of the communications stack. No changes are needed for existing applications. IPsec is the industry-standard network-security framework chosen by the IETF for both the IP Version 4 and 6 environments.

IPsec protects your data traffic using the following cryptographic techniques:

Authentication
Process by which the identity of a host or end point is verified

Integrity Checking
Process of ensuring that no modifications were made to the data while in transit across the network

Encryption
Process of ensuring privacy by ″hiding″ data and private IP addresses while in transit across the network

Authentication algorithms prove the identity of the sender and data integrity by using a cryptographic hash function to process a packet of data (with the fixed IP header fields included) using a secret key to produce a unique digest. On the receiver side, the data is processed using the same function and key. If either the data has been altered or the sender key is not valid, the datagram is discarded.

Encryption uses a cryptographic algorithm to modify and randomize the data using a certain algorithm and key to produce encrypted data known as *cyphertext*. Encryption makes the data unreadable while in transit. After it is received, the data is recovered using the same algorithm and key (with symmetric encryption algorithms). Encryption must occur with authentication to verify the data integrity of the encrypted data.

These basic services are implemented in IPsec by the use of the Encapsulating Security Payload (ESP) and the Authentication Header (AH). ESP provides confidentiality by encrypting the original IP packet, building an ESP header, and putting the cyphertext in the ESP payload.

The AH can be used alone for authentication and integrity-checking if confidentiality is not an issue. With AH, the static fields of the IP header and the data have a hash algorithm applied to compute a keyed digest. The receiver uses its key to compute and compare the digest to make sure the packet is unaltered and the sender’s identity is authenticated.

IP security features

This section provides the features of IP Security.

- Hardware acceleration with the 10/100 Mbps Ethernet PCI Adapter II.
- AH support using RFC 2402, and ESP support using RFC 2406.
- Manual tunnels can be configured to provide interoperability with other systems that do not support the automatic IKE key refreshment method, and for use of IP Version 6 tunnels.
- Tunnel mode and transport mode of encapsulation for host or gateway tunnels.
- Authentication algorithms of HMAC (Hashed Message Authentication Code) MD5 (Message Digest 5) and HMAC SHA (Secure Hash Algorithm).
- Encryption algorithms include 56 bit Data Encryption Standard (DES) Cipher Block Chaining (CBC) with 64 bit initial vector (IV), Triple DES, DES CBC 4 (32 bit IV).

- Dual IP Stack Support (IP version 4 and IP version 6).
- Both IP Version 4 and IP Version 6 traffic can be encapsulated and filtered. Because the IP stacks are separate, the IP Security function for each stack can be configured independently.
- Filtering of secure and nonsecure traffic by a variety of IP characteristics such as source and destination IP addresses, interface, protocol, port numbers, and more.
- Automatic filter-rule creation and deletion with most tunnel types.
- Use of host names for the destination address when defining tunnels and filter rules. The host names are converted to IP addresses automatically (as long as DNS is available).
- Logging of IP Security events to **syslog**.
- Use of system traces and statistics for problem determination.
- User-defined default action allows the user to specify whether traffic that does not match defined tunnels should be allowed.

Internet Key Exchange features:

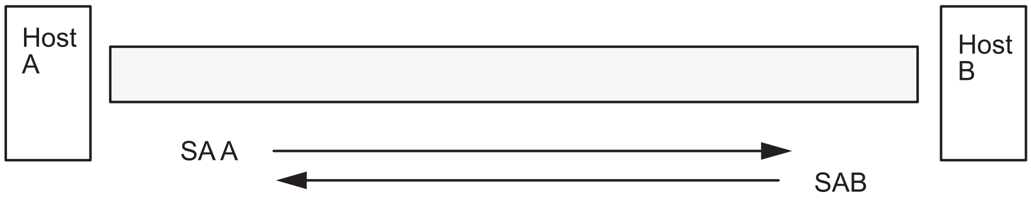
The following features are available with Internet Key Exchange (beginning with AIX 4.3.3):

- Support incoming PKCS #7 certificate chains
- Certificate Revocation List support with retrieval using HTTP or LDAP servers.
- Automatic key refreshment with tunnels using IETF Internet Key Exchange (IKE) protocol.
- X.509 Digital Certificate and preshared key support in IKE protocol during key negotiation.
- IKE tunnels can be created using Linux configuration files (AIX 5.1 and later).
- Authentication with preshared keys and X.509 digital signatures.
- Use of main mode (identity protect mode) and aggressive mode.
- Support for Diffie Hellman groups 1, 2, and 5.
- ESP encryption support for Data Encryption Standard (DES), Triple DES, Null encryption; ESP authentication support with HMAC MD5 and HMAC SHA1.
- AH support for HMAC MD5 and HMAC SHA1.
- IP Version 4 and Version 6 support.

Security associations

The building block on which secure communications is built is a concept known as a *security association*. Security associations relate a specific set of security parameters to a type of traffic.

With data protected by IP Security, a separate security association exists for each direction and for each header type, AH or ESP. The information contained in the security association includes the IP addresses of the communicating parties, a unique identifier known as the Security Parameters Index (SPI), the algorithms selected for authentication or encryption, the authentication and encryption keys, and the key lifetimes. The following figure shows the security associations between Host A and Host B.



SA = Security Association, consisting of:

Destination address
SPI
Key
Crypto Algorithm and Format
Authentication Algorithm
Key Lifetime

Figure 6. Establishment of a Secure Tunnel Between Hosts A and B

This illustration shows a virtual tunnel running between Host A and Host B. Security association A is an arrow directed from Host A to Host B. Security association B is an arrow directed from Host B to Host A. A Security association consists of the Destination Address, SPI, Key, Crypto Algorithm and Format, Authentication Algorithm, and Key Lifetime.

The goal of key management is to negotiate and compute the security associations that protect IP traffic.

Tunnels and key management

Use a tunnel to negotiate and manage the security associations that are required to set up secure communication between two hosts.

The following types of tunnels are supported, each using a different key management technique:

- IKE tunnels (dynamically changing keys, IETF standard)
- Manual tunnels (static, persistent keys, IETF standard)

Internet Key Exchange tunnel support:

IKE Tunnels are based on the Internet Security Association and Key Management Protocol (ISAKMP)/Oakley standards developed by the IETF. With this protocol, security parameters are negotiated and refreshed, and keys are exchanged securely.

The following types of authentication are supported: preshared key and X.509v3 digital certificate signatures.

The negotiation uses a two-phase approach. The first phase authenticates the communicating parties, and specifies the algorithms to be used for securely communicating in phase 2. During phase 2, IP Security parameters to be used during data transfer are negotiated, security associations and keys are created and exchanged.

The following table shows the authentication algorithms that can be used with the AH and ESP security protocols for IKE tunnel support.

Algorithm	**AH IP Version 4 & 6**	**ESP IP Version 4 & 6**
HMAC MD5	X	X
HMAC SHA1	X	X
DES CBC 8		X

Algorithm	AH IP Version 4 & 6	ESP IP Version 4 & 6
Triple DES CBC		X
ESP Null		X

Manual tunnel support:

Manual tunnels provide backward compatibility, and they interoperate with machines that do not support IKE key management protocols. The disadvantage of manual tunnels is that the key values are static. The encryption and authentication keys are the same for the life of the tunnel and must be manually updated.

The following table shows the authentication algorithms that can be used with the AH and ESP security protocols for manual tunnel support.

Algorithm	AH IP Version 4	AH IP Version 6	ESP IP Version 4	ESP IP Version 6
HMAC MD5	X	X	X	X
HMAC SHA1	X	X	X	X
Triple DES CBC			X	X
DES CBC 8			X	X
DES CBC 4			X	X

Because IKE tunnels offer more effective security, IKE is the preferred key management method.

Native filtering capability

Filtering is a basic function in which incoming and outgoing packets can be accepted or denied based on a variety of characteristics. This allows a user or system administrator to configure the host to control the traffic between this host and other hosts.

Filtering is done on a variety of packet properties, such as source and destination addresses, IP version (4 or 6), subnet masks, protocol, port, routing characteristics, fragmentation, interface, and tunnel definition.

Rules, known as *filter rules*, are used to associate certain kinds of traffic with a particular tunnel. In a basic configuration for manual tunnels, when a user defines a host-to-host tunnel, filter rules are autogenerated to direct all traffic from that host through the secure tunnel. If more specific types of traffic are desired (for instance, subnet to subnet), the filter rules can be edited or replaced to allow precise control of the traffic using a particular tunnel.

For IKE tunnels, the filter rules are also automatically generated and inserted in the filter table once the tunnel is activated.

Similarly, when the tunnel is modified or deleted, the filter rules for that tunnel are automatically deleted, which simplifies IP Security configuration and helps reduce human error. Tunnel definitions can be propagated and shared among machines and firewalls using import and export utilities, which is helpful in the administration of a large number of machines.

Filter rules associate particular types of traffic with a tunnel, but data being filtered does not necessarily need to travel in a tunnel. This aspect of filter rules lets the operating system provide basic firewall functionality to those who want to restrict traffic to or from their machine in an intranet or in a network that does not have the protection of a true firewall. In this scenario, filter rules provide a second barrier of protection around a group of machines.

After the filter rules are generated, they are stored in a table and loaded into the kernel. When packets are ready to be sent or received from the network, the filter rules are checked in the list from top to bottom to determine whether the packet should be permitted, denied, or sent through a tunnel. The criteria of the rule is compared to the packet characteristics until a match is found or the default rule is reached.

The IP Security function also implements filtering of non-secure packets based on very granular, user-defined criteria, which allows the control of IP traffic between networks and machines that do not require the authentication or encryption properties of IP Security.

Digital certificate support

IP Security supports the use of X.509 Version 3 digital certificates.

The Key Manager tool manages certificate requests, maintains the key database, and performs other administrative functions.

Digital certificates are described in Digital Certificate Configuration. The Key Manager and its functions are described in Using the IBM Key Manager Tool

Virtual private networks and IP security

A virtual private network (VPN) securely extends a private intranet across a public network such as the Internet.

VPNs convey information across what is essentially a private tunnel through the Internet to and from remote users, branch offices, and business partners/suppliers. Companies can opt for Internet access through Internet service providers (ISPs) using direct lines or local telephone numbers and eliminate more expensive leased lines, long-distance calls, and toll-free telephone numbers. A VPN solution can use the IPsec security standard because IPsec is the IETF-chosen industry standard network security framework for both the IP Version 4 and 6 environments, and no changes are needed for existing applications.

A recommended resource for planning the implementation of a VPN in AIX is Chapter 9 of *A Comprehensive Guide to Virtual Private Networks, Volume III: Cross-Platform Key and Policy Management*, ISBN SG24-5309-00. This guide is also available on the Internet World Wide Web at http://www.redbooks.ibm.com/redbooks/SG245309.html.

Installing the IP security feature

The IP Security feature in AIX is separately installable and loadable.

The file sets that need to be installed are as follows:

- **bos.net.ipsec.rte** (The run-time environment for the kernel IP Security environment and commands)
- **bos.msg.***LANG***.net.ipsec** (where *LANG* is the desired language, such as en_US)
- **bos.net.ipsec.keymgt**
- **bos.net.ipsec.websm**
- **bos.crypto-priv** (The file set for DES and triple DES encryption)

The **bos.crypto-priv** file set is located on the Expansion Pack. For IKE digital signature support, you must also install the **gskit.rte** fileset (AIX Version 4) or **gskkm.rte** (AIX 5.1) from the Expansion Pack.

For IP Security support in Web-based System Manager, you must install the **Java131.ext.xml4j** fileset at level 1.3.1.1 or later.

After it is installed, IP Security can be separately loaded for IP Version 4 and IP Version 6, either by using the recommended procedure provided in “Loading IP security” on page 144 or by using the **mkdev** command.

Loading IP security

Use SMIT or Web-based System Manager to automatically load the IP security modules when IP Security is started. Also, SMIT and Web-based System Manager ensure that the kernel extensions and IKE daemons are loaded in the correct order.

Note: Loading IP Security enables the filtering function. Before loading, it is important to ensure the correct filter rules are created. Otherwise, all outside communication might be blocked.

If the loading completes successfully, the **lsdev** command shows the IP Security devices as `Available`.

```
lsdev -C -c ipsec

   ipsec_v4 Available IP Version 4 Security Extension
   ipsec_v6 Available IP Version 6 Security Extension
```

After the IP Security kernel extension has been loaded, tunnels and filters are ready to be configured.

Planning IP security configuration

To configure IP Security, plan to configure the tunnels and filters first.

When a simple tunnel is defined for all traffic to use, the filter rules can be automatically generated. If more complex filtering is desired, filter rules can be configured separately.

You can configure IP Security using the Web-based System Manager Network plug-in, Virtual Private Network plug-in, or the System Management Interface Tool (SMIT). If using SMIT, the following fast paths are available:

smit ips4_basic
Basic configuration for IP version 4

smit ips6_basic
Basic configuration for IP version 6

Before configuring IP Security for your site, you must decide what method you intend to use; for example, whether you prefer to use tunnels or filters (or both), which type of tunnel best suits your needs, and so on. The following sections provide information you must understand before making these decisions:

- Hardware acceleration
- Tunnels versus filters
- Tunnels and security associations
- Choosing a tunnel type
- Using IKE with DHCP or dynamically-assigned addresses

Hardware acceleration

The 10/100 Mbps Ethernet PCI Adapter II (Feature code 4962) offers standards-based IP Security and is designed to offload IP Security functions from the AIX operating system.

When the 10/100 Mbps Ethernet PCI Adapter II is present in the AIX system, the IP Security stack uses the following capabilities of the adapter:

- Encryption and decryption using DES or Triple DES algorithms
- Authentication using the MD5 or SHA-1 algorithms
- Storage of the security-association information

The functions on the adapter are used instead of the software algorithms. The 10/100 Mbps Ethernet PCI Adapter II is available for manual and IKE tunnels.

The IP Security hardware acceleration feature is available in the 5.1.0.25 or later level of the **bos.net.ipsec.rte** and **devices.pci.1410ff01.rte** file sets.

There is a limit on the number of security associations that can be offloaded to the network adapter on the receive side (inbound traffic). On the transmit side (outbound traffic), all packets that use a supported configuration are offloaded to the adapter. Some tunnel configurations can not be offloaded to the adapter.

The 10/100 Mbps Ethernet PCI Adapter II supports the following features:

- DES, 3DES, or NULL encryption through ESP
- HMAC-MD5 or HMAC-SHA-1 authentication through ESP or AH, but not both. (If ESP and AH both used, ESP must be performed first. This is always true for IKE tunnels, but the user can select the order for manual tunnels.)
- Transport and Tunnel mode
- Offload of IPV4 packets

Note: The 10/100 Mbps Ethernet PCI Adapter II cannot handle packets with IP options.

To enable the 10/100 Mbps Ethernet PCI Adapter II for IP Security, you may have to detach the network interface and then enable the IPsec Offload feature.

To detach the network interface, perform the following steps using the SMIT interface:

To enable the IPsec Offload feature, do the following using the SMIT interface:

1. Login as the **root** user.
2. Type `smitty eadap` at the command line and press Enter.
3. Select the **Change / Show Characteristics of an Ethernet Adapter** option and press Enter.
4. Select the 10/100 Mbps Ethernet PCI Adapter II and press Enter.
5. Change the IPsec Offload field to yes and press Enter.

To detach the network interface, from the command line, type the following:

```
# ifconfig enX detach
```

To enable the IPsec offload attribute, from the command line, type the following:

```
# chdev -l entX -a ipsec_offload=yes
```

To verify that the IPsec offload attribute has been enabled, from the command line, type the following:

```
# lsattr -El entX detach
```

To disable the IPsec offload attribute, from the command line, type the following:

```
# chdev -l entX -a ipsec_offload=no
```

Use the **enstat** command to ensure that your tunnel configuration is taking advantage of the IPsec offload attribute. The **enstat** command shows all the statistics of transmit and receive IPsec packets when the IPsec offload attribute is enabled. For example, if the Ethernet interface is **ent1**, type the following:

```
# entstat -d ent1
```

The output will be similar to the following example:

```
.
.
.
10/100 Mbps Ethernet PCI Adapter II (1410ff01) Specific Statistics:
--------------------------------------------
.
.
```

```
.
Transmit IPsec packets: 3
Transmit IPsec packets dropped: 0
Receive IPsec packets: 2
Receive IPsec packets dropped: 0
```

Tunnels versus filters

Two distinct parts of IP Security are *tunnels* and *filters*. Tunnels require filters, but filters do not require tunnels.

Filtering is a function in which incoming and outgoing packets can be accepted or denied based on a variety of characteristics called *rules*. This function allows a system administrator to configure the host to control the traffic between this host and other hosts. Filtering is done on a variety of packet properties, such as source and destination addresses, IP Version (4 or 6), subnet masks, protocol, port, routing characteristics, fragmentation, interface, and tunnel definition. This filtering is done at the IP layer, so no changes are required to the applications.

Tunnels define a security association between two hosts. These security associations involve specific security parameters that are shared between end points of the tunnel.

The following illustration indicates how a packet comes in from the network adapter to the IP stack. From there, the filter module is called to determine if the packet is permitted or denied. If a tunnel ID is specified, the packet is checked against the existing tunnel definitions. If the decapsulation from the tunnel is successful, the packet is passed to the upper-layer protocol. This function occurs in reverse order for outgoing packets. The tunnel relies on a filter rule to associate the packet with a particular tunnel, but the filtering function can occur without passing the packet to the tunnel.

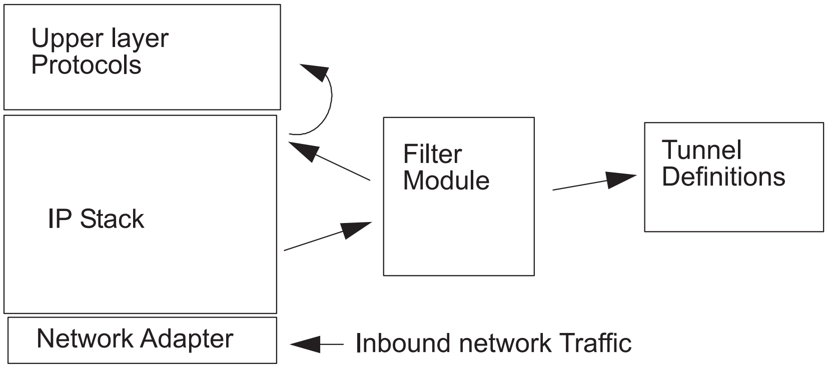


Figure 7. Network Packet Routing

The illustration shows the route a network packet takes. Inbound from the network, the packet enters the network adapter. from there it goes to the IP stack where it is sent to the filter module. From the filter module, the packet is either sent to tunnel definitions or it is returned to the IP stack where it is forwarded to the upper-layer protocols.

Tunnels and security associations

Tunnels are used whenever you need to have data authenticated, or authenticated and encrypted. Tunnels are defined by specifying a security association between two hosts. The security association defines the parameters for the encryption and authentication algorithms and characteristics of the tunnel.

The following illustration shows a virtual tunnel between Host A and Host B.

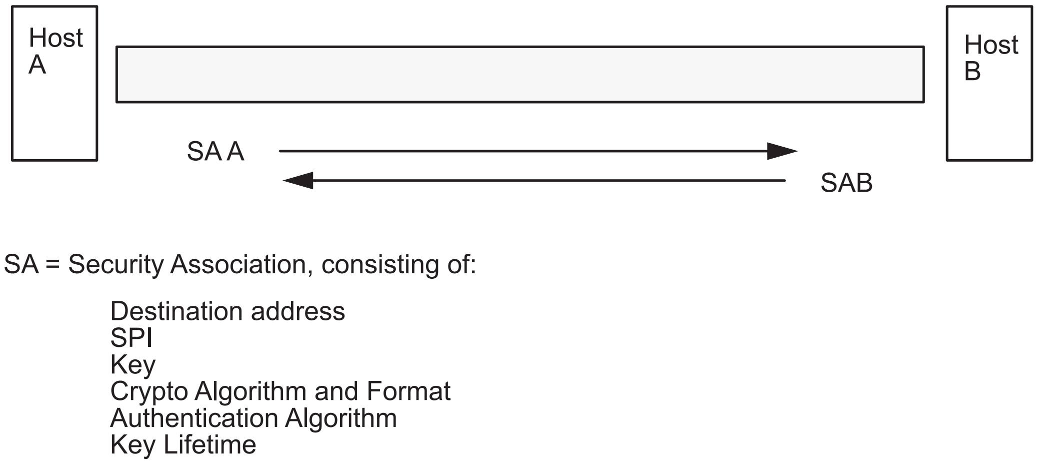


Figure 8. Establishment of a Secure Tunnel Between Hosts A and B

The illustration shows a virtual tunnel running between Host A and Host B. Security association A is an arrow directed from Host A to Host B. Security association B is an arrow directed from Host B to Host A. A security association consists of the Destination Address, SPI, Key, Crypto Algorithm and Format, Authentication Algorithm, and Key Lifetime.

The Security Parameter Index (SPI) and the destination address identify a unique security association. These parameters are required for uniquely specifying a tunnel. Other parameters such as cryptographic algorithm, authentication algorithm, keys, and lifetime can be specified or defaults can be used.

Tunnel considerations

You should consider several things before deciding which type of tunnel to use for IP security.

IKE tunnels differ from manual tunnels because the configuration of security policies is a separate process from defining tunnel endpoints.

In IKE, there is a two-step negotiation process. Each step of the negotiation process is called a *phase*, and each phase can have separate security policies.

When the Internet Key negotiation starts, it must set up a secure channel for the negotiations. This is known as the *key management* phase or *phase 1*. During this phase, each party uses preshared keys or digital certificates to authenticate the other and pass ID information. This phase sets up a security association during which the two parties determine how they plan to communicate securely and then which protections are to be used to communicate during the second phase. The result of this phase is an *IKE* or *phase 1* tunnel.

The second phase is known as the *data management* phase or *phase 2* and uses the IKE tunnel to create the security associations for AH and ESP that actually protect traffic. The second phase also determines the data that will be using the IP Security tunnel. For example, it can specify the following:

- A subnet mask
- An address range
- A protocol and port number combination

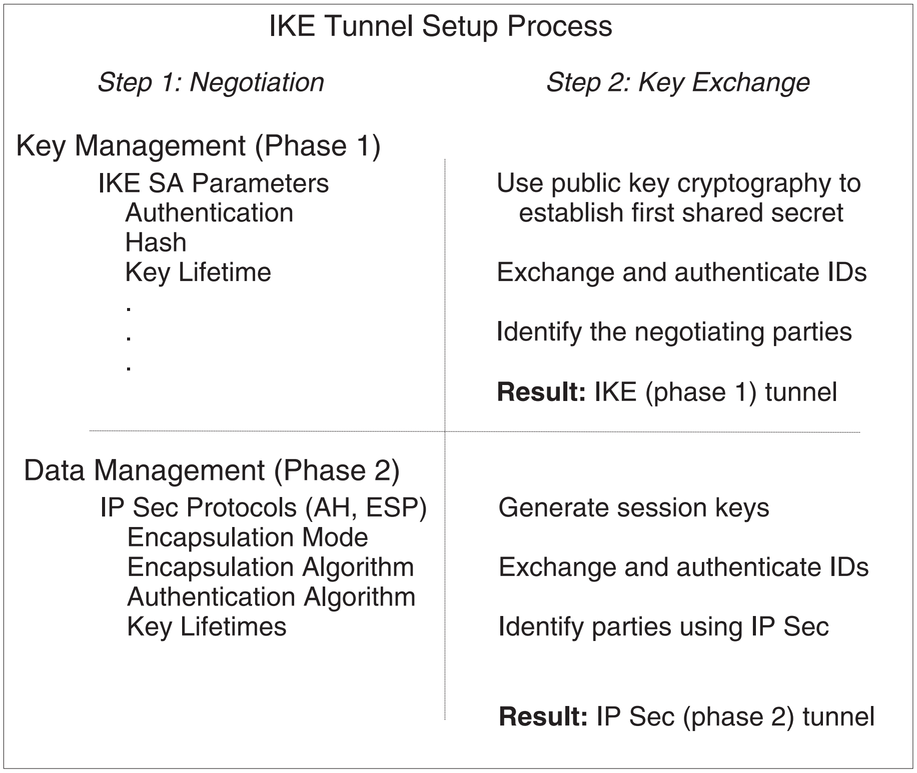

IKE Tunnel Setup Process
Step 1: Negotiation
Step 2: Key Exchange
Key Management (Phase 1)
IKE SA Parameters
Authentication
Hash
Key Lifetime
.
.
.
Use public key cryptography to establish first shared secret
Exchange and authenticate IDs
Identify the negotiating parties
Result: IKE (phase 1) tunnel
Data Management (Phase 2)
IP Sec Protocols (AH, ESP)
Encapsulation Mode
Encapsulation Algorithm
Authentication Algorithm
Key Lifetimes
Generate session keys
Exchange and authenticate IDs
Identify parties using IP Sec
Result: IP Sec (phase 2) tunnel

Figure 9. IKE Tunnel Setup Process

This illustration shows the two-step, two-phase process for setting up an IKE tunnel.

In many cases, the endpoints of the key management (IKE) tunnel will be the same as the endpoints of the data management (IP Security) tunnel. The IKE tunnel endpoints are the IDs of the machines carrying out the negotiation. The IP Security tunnel endpoints describe the type of traffic that will use the IP Security tunnel. For simple host-to-host tunnels, in which all traffic between two tunnels is protected with the same tunnel, the phase 1 and phase 2 tunnel endpoints are the same. When negotiating parties are two gateways, the IKE tunnel endpoints are the two gateways, and the IP Security tunnel endpoints are the machines or subnets (behind the gateways) or the range of addresses (behind the gateways) of the tunnel users.

Key management parameters and policy:

You can customize key-management policy by specifying the parameters to be used during IKE negotiation. For example, there are key-management policies for pre-shared key or signature mode authentication. For Phase 1, the user must determine certain key-management security properties with which to carry out the exchange.

Phase 1 (the key management phase) sets the following parameters of an IKE tunnel configuration as shown in the table below:

Key Management (Phase 1) Tunnel	Name of this IKE tunnel. For each tunnel, the endpoints of the negotiation must be specified. These are the two machines that plan to send and validate IKE messages. The name of the tunnel may describe the tunnel endpoints such as `VPN Boston` or `VPN Acme`.

Host Identity Type	ID type that will be used in the IKE exchange. The ID type and value must match the value for the preshared key to ensure that proper key lookup is performed. If a separate ID is used to search a preshared key value, the *host ID* is the key's ID and its *type* is KEY_ID. The KEY_ID type is useful if a single host has more than one preshared key value.
Host Identity	Value of the host ID represented as an IP address, a fully qualified domain name (FQDN), or a user at the fully qualified domain name (*user@FQDN*). For example, jdoe@studentmail.ut.edu.
IP Address	IP address of the remote host. This value is required when the host ID type is KEY_ID or whenever the host ID type cannot be resolved to an IP address. For example, if the user name cannot be resolved with a local name server, the IP address for the remote side must be entered.

Data management parameters and policy:

The data management proposal parameters are set during phase 2 of an IKE tunnel configuration. They are the same IP Security parameters used in manual tunnels and describe the type of protection to be used for protecting data traffic in the tunnel. You can start more than one phase 2 tunnel under the same phase 1 tunnel.

The following endpoint ID types describe the type of data that uses the IP Security Data tunnel:

Host, **Subnet**, or **Range**	Describes whether the data traffic traveling in the tunnel will be for a particular host, subnet, or address range.
Host/Subnet ID	Contains the host or subnet identity of the local and remote systems passing traffic over this tunnel. Determines the IDs sent in the phase 2 negotiation and the filter rules that will be built if the negotiation is successful.
Subnet mask	Describes all IP addresses within the subnet (for example, host 9.53.250.96 and mask 255.255.255.0)
Starting IP Address Range	Provides the starting IP address for the range of addresses that will be using the tunnel (for example, 9.53.250.96 of 9.53.250.96 to 9.53.250.93)
Ending IP Address Range	Provides the ending IP address for the range of addresses that will be using the tunnel (for example, 9.53.250.93 of 9.53.250.96 to 9.53.250.93)
Port	Describes data using a specific port number (for example, 21 or 23)
Protocol	Describes data being transported with a specific protocol (for example, TCP or UDP). Determines the protocol sent in the phase 2 negotiation and the filter rules that will be built if the negotiation is successful. The protocol for the local endpoint must match the protocol for the remote end point.

Choosing a tunnel type:

The decision to use manual tunnels or IKE tunnels depends on the tunnel support of the remote end and the type of key management desired.

When available, use IKE tunnels because they offer industry-standard secure key negotiation and key refreshment. They also take advantage of the IETF ESP and AH header types and support anti-replay protection. You can optionally configure signature mode to allow digital certificates.

If the remote end uses one of the algorithms requiring manual tunnels, manual tunnels should be used. Manual tunnels ensure interoperability with a large number of hosts. Because the keys are static and difficult to change and might be cumbersome to update, they are not as secure. Manual tunnels can be used between a host running this operating system and any other machine running IP Security and having a common set of cryptographic and authentication algorithms. Most vendors offer Keyed MD5 with DES, or HMAC MD5 with DES. This subset works with almost all implementations of IP Security.

The procedure used in setting up manual tunnels, depends on whether you are setting up the first host of the tunnel or setting up the second host, which must have parameters matching the first host setup. When setting up the first host, the keys can be autogenerated, and the algorithms can be defaulted. When setting up the second host, import the tunnel information from the remote end, if possible.

Another important consideration is determining whether the remote system is behind a firewall. If it is, the setup must include information about the intervening firewall.

Using IKE with DHCP or dynamically assigned addresses

One common scenario for using IP Security with an operating system is when remote systems are initiating IKE sessions with a server, and their identity cannot be tied to a particular IP address.

This case can occur in a Local Area Network (LAN) environment such as using IP Security to connect to a server on a LAN and wanting to encrypt the data. Other common uses involve remote clients dialing into a server and using either a fully qualified domain name (FQDN), or e-mail address (user@FQDN) to identify the remote ID.

In the Key Management phase (Phase 1), an RSA Signature is the only authentication mode supported if you use main mode with non-IP address IDs. In another words, if you want use pre-shared key authentication, you must use aggressive mode or main mode with IP addresses as IDs. In fact, when the number of DHCP clients with whom you want establish IPsec tunnels is large, it becomes impractical to define unique, pre-shared keys for each DHCP client, so it is recommend you use RSA Signature authentication in this scenario. You also can use Group ID as a remote ID in tunnel definition so that you only define the tunnel once with all DHCP clients (see tunnel definition sample file **/usr/samples/ipsec/group_aix_responder.xml**). Group ID is a unique feature of AIX IPsec. You can define a group ID to include any IKE IDs (like a single IP address), FQDN, User FQDN, a range or set of IP addresses, and so on, and then use this Group ID as the phase 1 or phase 2 remote ID in your tunnel definitions.

Note: When Group ID is used, tunnel should be defined as Responder role only. That means you must activate this tunnel from the DHCP client side.

For the Data Management phase (Phase 2), when the IP Security associations are being created to encrypt TCP or UDP traffic, a generic data management tunnel can be configured. Therefore, any request that was authenticated during phase 1, will use the generic tunnel for defined Data Management phase if the IP address is not explicitly configured in the database. This allows any address to match the generic tunnel and can be used as long as the rigorous public key-based security validation was successful in phase 1.

Using XML to define a generic data management tunnel:

You can define a generic Data Management tunnel using the XML format understood by **ikedb**.

See the section entitled “Command-line interface for IKE tunnel configuration” on page 155 for more information on the IKE XML interface and the **ikedb** command. Generic Data Management tunnels are used with DHCP. The XML format uses the tag name IPSecTunnel for what Web-based System Manager calls a Data Management tunnel. This is also referred to as a *phase 2 tunnel* in other contexts. A *generic Data Management tunnel* is not a true tunnel, but an IPSecProtection that is used if an incoming Data Management message (under a specific Key Management tunnel) does not match any Data Management tunnel defined for that Key Management tunnel. It is only used in the case where the AIX system is the responder. Specifying a generic Data Management tunnel IPSecProtection is optional.

The generic Data Management tunnel is defined in the IKEProtection element. There are two XML attributes, called *IKE_IPSecDefaultProtectionRef* and *IKE_IPSecDefaultAllowedTypes*, that are used for this.

First, you need to define an IPSecProtection that you would like to use as the default if no IPSecTunnels (Data Management tunnels) match. An IPSecProtection that is to be used as a default must have an IPSec_ProtectionName that begins with `_defIPSprot_`.

Now go to the IKEProtection that you would like to use this default IPSecProtection. Specify an **IKE_IPSecDefaultProtectionRef** attribute that contains the name of the default IPSec_Protection.

You must also specify a value for the **IKE_IPSecDefaultAllowedTypes** attribute in this IKEProtection. It can have one or more of the following values (if multiple values, they should be space-separated):

```
Local_IPV4_Address
Local_IPV6_Address
Local_IPV4_Subnet
Local_IPV6_Subnet
Local_IPV4_Address_Range
Local_IPV6_Address_Range
Remote_IPV4_Address
Remote_IPV6_Address
Remote_IPV4_Subnet
Remote_IPV6_Subnet
Remote_IPV4_Address_Range
Remote_IPV6_Address_Range
```

These values correspond to the ID types specified by the initiator. In the IKE negotiation, the actual IDs are ignored. The specified IPSecProtection is used if the **IKE_IPSecDefaultAllowedTypes** attribute contains a string beginning with `Local_` that corresponds to the initiator's local ID type, and contains a string beginning with `Remote_` that corresponds to the initiator's remote ID type. In other words, you must have at least one `Local_` value and at least one `Remote_` value in any **IKE_IPSecDefaultAllowedTypes** attribute in order for the corresponding IPSec_Protection to be used.

General data management tunnel example:

An initiator sends the following to the AIX system in a phase 2 (Data Management) message:

```
local ID type:    IPV4_Address
local ID:         192.168.100.104

remote ID type:   IPV4_Subnet
remote ID:        10.10.10.2
remote netmask:   255.255.255.192
```

The AIX system does not have a Data Management tunnel matching these IDs. But it does have an IPSecProtection with the following attributes defined:

```
IKE_IPSecDefaultProtectionRef="_defIPSprot_protection4"
IKE_IPSecDefaultAllowedTypes="Local_IPV4_Address
                              Remote_IPV4_Address
                              Remote_IPV4_Subnet
                              Remote_IPV4_Address_Range"
```

The local ID type of the incoming message, IPV4_Address, matches one of the Local_ values of the allowed types, Local_IPV4_Address. Also, the remote ID of the message, IPV4_Subnet, matches the value Remote_IPV4_Subnet. Therefore the Data Management tunnel negotiation will proceed with `_defIPSprot_protection4` as the IPSecProtection.

The **/usr/samples/ipsec/default_p2_policy.xml** file is a full XML file defining a generic IPSecProtection that can be used as an example.

Using Web-based system manager to define a generic data management tunnel:

To define a generic Data Management tunnel using the Web-based System Manager interface do the following:

1. Select a Key Management tunnel in the IKE Tunnels container, and select the action to Define a Data Management Tunnel.
2. Select generic Data Management tunnel. The configuration panels are similar to the panels used to define a Data Management tunnel. However, the choices for the ID types are different. Explicit IDs do not need to be specified. The ID types, which are IP V4 or V6 Address Only, IP V4 or V6 Subnet Only, and IP V4 or V6 Address or Subnet, cover all allowable cases of IDs.
3. Set the rest of the information the same way as in a Data Management Tunnel and click OK. Each Key Management tunnel can only have one associated Generic Tunnel.

Note: A generic data management tunnel can *only* be used in the case where the AIX system is the responder.

Configuring Internet key exchange tunnels

This section provides information on how to configure Internet Key Exchange (IKE) tunnels using the Web-based System Manager interface, the System Management Interface Tool (SMIT), or the command line.

Using Web-based system manager to configure IKE tunnels

Web-based system manager can be used to configure IKE tunnels.

The “Using the basic configuration wizard” provides an easy way to define an IKE tunnel with pre-shared keys. For more advanced options, see “Advanced IKE tunnel configuration” on page 153.

Using the basic configuration wizard:

You can use the the basic configuration wizard to define an IKE tunnel through Web-based System Manager using pre-shared keys or certificates as the authentication method.

The Web-based System Manager adds new key management and data management IKE tunnels to the IP Security subsystem, allows you to input minimal data and choose some options, and makes use of common default values for such parameters as tunnel lifetime.

When using the basic configuration wizard, keep the following in mind:

- The wizard can be used only for initial tunnel configuration. To modify, delete, or activate a tunnel, use the IKE Tunnel plug-in or task bar.
- The tunnel name must be unique on your system, but you can use the same name on the remote system. For example, on the local and remote systems, the tunnel name can be *hostA_to_hostB*, but the Local IP Address and the Remote IP Address fields (endpoints) are switched.
- Phase 1 and phase 2 tunnels are defined with the same encryption and authentication algorithms.
- The preshared key must be entered in hexadecimal form (without a leading `0x`) or ASCII text.
- If digital certificates are chosen as the authentication method, you must use the Key Manager tool to create a digital certificate.
- The host ID type can only be `IP Address`.
- The transforms and proposals you create are assigned names that end with the user-defined tunnel name. You can view the transforms and proposals in Web-based System Manager through **VPN** and the **IKE Tunnel** plug-in.

Use the following procedure to configure a new tunnel using the wizard:

1. Open Web-based System Manager using the **wsm** command from the command line.

2. Select the Network plug-in.
3. Select **Virtual Private Networks (IP Security)**.
4. From the Console area, select the **Overview and Tasks** folder.
5. Select **Configure a Basic Tunnel Configuration wizard**.
6. Click on **Next** on the Step 1 Introduction panel, and then follow the steps to configure an IKE tunnel.

   Online help is available by pressing F1 if you need it.

   After a tunnel is defined using the wizard, the tunnel definition displays in the Web-based System Manager IKE tunnels list and can be activated or modified.

Advanced IKE tunnel configuration:

You can configure key management and data management tunnels separately, using the following procedures.

Configuring key management tunnels:

IKE tunnels are configured using Web-based System Manager.

Perform the following steps to add a key management tunnel:

1. Open Web-based System Manager using the **wsm** command.
2. Select the Network plug-in.
3. Select **Virtual Private Networks (IP Security)**.
4. From the Console area, select **Overview and Tasks**.
5. Select **Start IP Security**. This action loads the IP Security kernel extensions and starts the **isakmpd**, **tmd**, and **cpsd** daemons.

   A tunnel is created by defining the key management and data management endpoints and their associated security transforms and proposals.

   - Key management is the authentication phase. It sets up a secure channel between the negotiating parties needed before the final IP Security parameters and keys are computed.
   - Data management describes the type of traffic that will be using a particular tunnel. It can be configured for a single host or group of hosts (with the use of subnets or IP ranges) along with specified protocol and port numbers.

   The same key management tunnel can be used to protect multiple data management negotiations and key refreshes, as long as they take place between the same two endpoints; for example, between two gateways.
6. To define the key management tunnel endpoints, click **Internet Key Exchange (IKE) Tunnels** on the Identification tab.
7. Enter information to describe the identities of the systems taking part in the negotiations. In most cases, IP addresses are used, and a policy compatible with the remote side must be created.

   On the Transforms tab, use matching transforms on both sides, or contact the administrator on the remote end to define a matching transform. A transform containing several choices can be created to allow flexibility when proposing or matching on a transform.
8. If using preshared keys for authentication, enter the preshared key under the **key** tab. This value must match on both the remote and local machines.
9. Create a transform to be associated with this tunnel by clicking **Add** on the Transforms tab.

   To enable digital certificates and signature mode support, choose an authentication method of RSA Signature or RSA Signature with CRL Checking.

   For more information about digital certificates, see “Digital certificates and the key manager concepts” on page 158.

Configuring data management tunnels:

To complete IKE tunnel setup, you need to configure data management tunnel endpoints and proposals.

Open Web-based System Manager, as described in “Creating IKE tunnels using digital certificates” on page 167. Perform the following steps to create a data management tunnel:

1. Select a key management tunnel and define any unique options. Most data management options can remain as defined by the default.
2. Specify endpoint types (such as IP address, subnet, or IP address range) under the Endpoints tab. You can select a port number and protocol or accept the default.
3. On the Proposals panel, you can create a new proposal by clicking the **Add** button or clicking **OK** to create a proposal. If there are multiple proposals, you can use the Move Up or Move Down buttons to change the search order.

Group support:

IP security supports grouping IKE IDs in a tunnel definition to associate multiple IDs with a single security policy without having to create separate tunnel definitions.

Grouping is especially useful when setting up connections to several remote hosts, because you can avoid setting up or managing multiple tunnel definitions. Also, if changes must be made to a security policy, you do not need to change multiple tunnel definitions.

A group must be defined before using that group name in a tunnel definition. The group’s size is limited to 1 KB. On the initiator’s side of the negotiation, you can use groups as a remote ID in data management tunnel definitions only. On the responders side of the negotiation, you can use groups as a remote ID in key management and data management tunnel definitions.

A group is composed of a group name and a list of IKE IDs and ID types. IDs can be the same type or a mix of the following:

- IPv4 addresses
- IPv6 addresses
- FQDN
- user@FQDN
- X500 DN types

During a Security Association negotiation, the IDs in a group are searched linearly for the first match.

Web-based System Manager can be used to define a group that is to be used for the remote endpoint of a Key Management tunnel. To define a group using Web-based System Manager, perform the following steps:

1. Select a Key Management tunnel in the IKE Tunnel container.
2. Open the **Properties** dialog.
3. Select the **Identification** tab.
4. Select **group ID definition** for the remote host identity type.
5. Select the **Configure Group Definition** button and enter the group members in the window.

Refer to “Command-line interface for IKE tunnel configuration” on page 155 for information on defining groups from the command line.

Using the SMIT interface for IKE tunnel configuration

You can use the SMIT interface to configure IKE tunnels and perform basic IKE database functions.

SMIT uses underlying XML command functions to perform additions, deletions, and modifications to the IKE tunnel definitions. IKE SMIT is used in configuring IKE tunnels quickly and provides examples of the XML syntax used to create IKE tunnel definitions. The IKE SMIT menus also allow you to back up, restore, and initialize the IKE database.

To configure an IPv4 IKE tunnel, use the **smitty ike4** fast path. To configure an IPv6 IKE tunnel, use the **smitty ike6** fast path. The IKE database functions can be found in the Advanced IP Security Configuration menu.

All IKE database entries added through SMIT can be viewed or modified through the Web-based System Manager tool.

Command-line interface for IKE tunnel configuration

The **ikedb** command allows a user to retrieve, update, delete, import, and export information in the IKE database using an XML interface.

The **ikedb** command allows the user to write to (put) or read from (get) the IKE database. The input and output format is an Extensible Markup Language (XML) file. The format of an XML file is specified by its Document Type Definition (DTD). The **ikedb** command allows the user to see the DTD that is used to validate the XML file when doing a put. While entity declarations can be added to the DTD using the **-e** flag, this is the only modification to the DTD that can be made. Any external DOCTYPE declaration in the input XML file will be ignored and any internal DOCTYPE declaration might result in an error. The rules followed to parse the XML file using the DTD are specified in the XML standard. The **/usr/samples/ipsec** file has a sample of a typical XML file that defines common tunnel scenarios. See the **ikedb** command description in the *AIX 5L Version 5.3 Commands Reference* for syntax details.

You can use the **ike** command to start, stop, and monitor IKE tunnels. The **ike** command can also be used to activate, remove, or list IKE and IP Security tunnels. See the **ike** command description in the *AIX 5L Version 5.3 Commands Reference* for syntax details.

The following examples show how to use **ike**, **ikedb**, and several other commands to configure and check the status of your IKE tunnel:

1. To start a tunnel negotiation (*activate* a tunnel) or to allow the incoming system to act as a responder (depending on the role that is specified), use the **ike** command with a tunnel number, as follows:

   ```
   # ike cmd=activate numlist=1
   ```

   You can also use remote id or IP addresses, as shown in the following examples:

   ```
   # ike cmd=activate remid=9.3.97.256
   # ike cmd=activate ipaddr=9.3.97.100, 9.3.97.256
   ```

   Because it might take several seconds for the commands to complete, the command returns after the negotiation is started.
2. To display the tunnel status, use the **ike** command, as follows:

   ```
   # ike cmd=list
   ```

   The output looks similar to the following:

   ```
   Phase 1 Tunnel ID       [1]
   Phase 2 Tunnel ID       [1]
   ```

   The output shows phase 1 and phase 2 tunnels that are currently active.
3. To get a verbose listing of the tunnel, use the **ike** command, as follows:

   ```
   # ike cmd=list verbose
   ```

   The output looks similar to the following:

   ```
   Phase 1 Tunnel ID       1
   Local ID Type:          Fully_Qualified_Domain_Name
   Local ID:               bee.austin.ibm.com
   Remote ID Type:         Fully_Qualified_Domain_Name
   Remote ID:              ipsec.austin.ibm.com
   ```

```
Mode:                   Aggressive
Security Policy:        BOTH_AGGR_3DES_MD5
Role:                   Initiator
Encryption Alg:         3DES-CBC
Auth Alg:               Preshared Key
Hash Alg:               MD5
Key Lifetime:           28800 Seconds
Key Lifesize:           0 Kbytes
Key Rem Lifetime:       28737 Seconds
Key Rem Lifesize:       0 Kbytes
Key Refresh Overlap:    5%
Tunnel Lifetime:        2592000 Seconds
Tunnel Lifesize:        0 Kbytes
Tun Rem Lifetime:       2591937 Seconds
Status:                 Active

Phase 2 Tunnel ID       1
Local ID Type:          IPv4_Address
Local ID:               10.10.10.1
Local Subnet Mask:      N/A
Local Port:             any
Local Protocol:         all
Remote ID Type:         IPv4_Address
Remote ID:              10.10.10.4
Remote Subnet Mask:     N/A
Remote Port:            any
Remote Portocol:        all
Mode:                   Oakley_quick
Security Policy:        ESP_3DES_MD5_SHA_TUNNEL_NO_PFS
Role:                   Initiator
Encryption Alg:         ESP_3DES
AH Transform:           N/A
Auth Alg:               HMAC-MD5
PFS:                    No
SA Lifetime:            600 Seconds
SA Lifesize:            0 Kbytes
SA Rem Lifetime:        562 Seconds
SA Rem Lifesize:        0 Kbytes
Key Refresh Overlap:    15%
Tunnel Lifetime:        2592000 Seconds
Tunnel Lifesize:        0 Kbytes
Tun Rem Lifetime:       2591962 Seconds
Assoc P1 Tunnel:        0
Encap Mode:             ESP_tunnel
Status:                 Active
```

4. To display the filter rules in the dynamic filter table for the newly activated IKE tunnel, use the **lsfilt** command as follows:

   ```
   # lsfilt -d
   ```

   The output looks similar to the following example:

   ```
   1 permit 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 no udp eq 4001 eq 4001 both both no all
     packets 0 all
   2 *** Dynamic filter placement rule *** no
   0 permit 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 yes all any 0 any 0 both both no all
     packets 0 all

   *** Dynamic table ***

   0 permit 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 no udp eq 500 eq 500 local both no all
     packets 0
   0 permit 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 no ah any 0 any 0 both inbound no all
     packets 0
   0 permit 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 no esp any 0 any 0 both inbound no all
     packets 0
   ```

```
1 permit 10.10.10.1 255.255.255.255 10.10.10.4 255.255.255.255 no all any 0 any
  0 both outbound yes all packets 1
1 permit 10.10.10.4 255.255.255.255 10.10.10.1 255.255.255.255 no all any 0 any
  0 both inbound yes all packets 1
```

This example shows a machine that has one IKE tunnel and no other tunnels. The dynamic filter placement rule (rule #2 in this example output of the static table) can be moved by the user to control placement relative to all other user-defined rules. The rules in the dynamic table are constructed automatically as tunnels are negotiated and corresponding rules are inserted into the filter table. These rules can be displayed, but not edited.

5. To turn on logging of the dynamic filter rules, set the logging option for rule #2 to Yes, use the **chfilt** command, as shown in the following example:

   ```
   # chfilt -v 4 -n 2 -l y
   ```

   For more details on logging IKE traffic, see "Logging facilities" on page 178.
6. To deactivate the tunnel, use the **ike** command as follows:

   ```
   # ike cmd=remove numlist=1
   ```
7. To view tunnel definitions, use the **ikedb** command as follows:

   ```
   # ikedb -g
   ```
8. To put definitions to the IKE database from an XML file that has been generated on a peer machine and overwrite any existing objects in the database with the same name, use the **ikedb** command as follows:

   ```
   # ikedb -pFs peer_tunnel_conf.xml
   ```

   The **peer_tunnel_conf.xml** is the XML file generated on a peer machine.
9. To get the definition of the phase 1 tunnel named *tunnel_sys1_and_sys2* and all dependent phase 2 tunnels with respective proposals and protections, use the **ikedb** command, as follows:

   ```
   # ikedb -gr -t IKETunnel -n tunnel_sys1_and_sys2
   ```
10. To delete all preshared keys from the database, use the **ikedb** command, as follows:

    ```
    # ikedb -d -t IKEPresharedKey
    ```

For general information on IKE tunnel group support, see "Group support" on page 154. You can use the **ikedb** command to define groups from the command line.

AIX IKE and Linux affinity:

It is possible to configure an AIX IKE tunnel using Linux configuration files.

To configure an AIX IKE tunnel using Linux configuration files (AIX 5.1 and later), use the **ikedb** command with the **-c** flag (conversion option), which lets you use the **/etc/ipsec.conf** and **/etc/ipsec.secrets** Linux configuration files as IKE tunnel definitions. The **ikedb** command parses the Linux configuration files, creates an XML file, and optionally adds the XML tunnel definitions to the IKE database. You can then view the tunnel definitions by using either the **ikedb -g** command or the Web-based System Manager.

IKE tunnel configuration scenarios

The following scenarios describe the type of situations most customers encounter when trying to set up tunnels. These scenarios can be described as the branch office, business partner, and remote access cases.

- In the branch office case, the customer has two trusted networks that they want to connect together—the engineering group of one location to the engineering group of another. In this example, there are gateways that connect to each other and all the traffic passing between the gateways use the same tunnel. The traffic at either end of the tunnel is decapsulated and passes in the clear within the company intranet.

  In the first phase of the IKE negotiation, the IKE security association is created between the two gateways. The traffic that passes in the IP Security tunnel is the traffic between the two subnets, and the subnet IDs are used in the phase 2 negotiation. After the security policy and tunnel parameters are entered for the tunnel, a tunnel number is created. Use the **ike** command to start the tunnel.

- In the business partner scenario, the networks are not trusted, and the network administrator may want to restrict access to a smaller number of hosts behind the security gateway. In this case, the tunnel between the hosts carries traffic protected by IP Security for use between two particular hosts. The protocol of the phase 2 tunnel is AH or ESP. This host-to-host tunnel is secured within a gateway-to-gateway tunnel.
- In the remote access case, the tunnels are set up on demand and a high level of security is applied. The IP addresses may not be meaningful, therefore, fully qualified domain names or *user@* fully qualified domain names are preferred. Optionally, you can use KEYID to relate a key to a host ID.

Digital certificates and the key manager concepts

Digital certificates bind an identity to a public key, through which you can verify the sender or the recipient of an encrypted transfer.

Beginning with AIX 4.3.2, IP Security uses digital certificates to enable *public-key cryptography*, also known as *asymmetric cryptography*, which encrypts data using a private key known only to the user and decrypts it using an associated public (shared) key from a given public-private key pair. *Key pairs* are long strings of data that act as keys to a user's encryption scheme.

In public-key cryptography, the public key is given to anyone with whom the user wants to communicate. The sender digitally signs all secure communications with the corresponding private key for their assigned key pair. The recipient uses the public key to verify the sender's signature. If the message is successfully decrypted with the public key, the receiver can verify that the sender was authenticated.

Public-key cryptography relies on trusted, third parties, known as a *certification authorities* (*CAs*), to issue reliable digital certificates. The recipient specifies which issuing organizations or authorities are deemed trusted. A certificate is issued for a specific amount of time; when its expiration date has passed, it must be replaced.

AIX 4.3.2 and later versions provide the Key Manager tool, which manages digital certificates. The following sections provide conceptual information about the certificates themselves. Management tasks for these certificates are described in "Digital certificates and the key manager concepts."

Format of digital certificates

The digital certificate contains specific pieces of information about the identity of the certificate owner and about the certification authority. See the following figure for an illustration of a digital certificate.

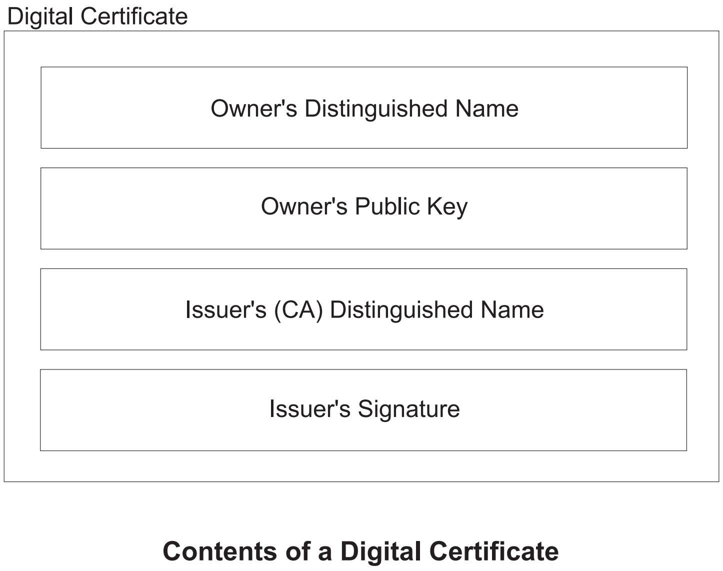

Digital Certificate
Owner's Distinguished Name
Owner's Public Key
Issuer's (CA) Distinguished Name
Issuer's Signature
Contents of a Digital Certificate

Figure 10. Contents of a Digital Certificate

This illustration shows the four entities of a digital certificate. From the top they are, Owner’s Distinguished Name, Owners Public Key, Issuer’s (CA) Distinguished Name, and Issuer’s Signature.

The following list further describes the contents of the digital certificate:

Owner’s Distinguished Name
Combination of the owner’s common name and context (position) in the directory tree. In the following figure of a simple directory tree, for example, Prasad is the owner’s common name and the context is country=US, organization=ABC, lower organization=SERV; therefore, the distinguished name is:

```
/C=US/O=ABC/OU=SERV/CN=prasad.austin.ibm.com
```

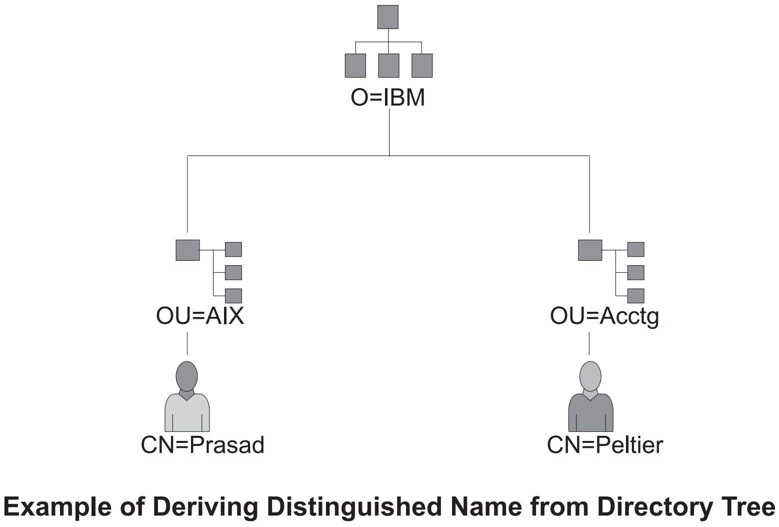


Figure 11. Example of Deriving Distinguished Name from Directory Tree

This illustration is a directory tree with O=ABC at the top level and branching to two entities on the second level. Level two contains OU=AIX and OU=Acctg on separate branches; each has a branch leading to a single entity on the last level. The last level contains CN=Prasad and CN=Peltier respectively.

Owner's Public Key
Used by the recipients to decrypt data.

Subject Alternate Name
Can be an identifier such as an IP address, e-mail address, fully qualified domain name, and so on.

Issue Date
Date the digital certificate was issued.

Expiration Date
Date the digital certificate expires.

Issuer's Distinguished Name
Distinguished name of the Certification Authority.

Issuer's Digital Signature
Digital signature used to validate a certificate.

Security considerations for digital certificates

A digital certificate alone cannot prove identity.

The digital certificate only allows you to verify the identity of the digital certificate owner by providing the public key that is needed to check the owner's digital signature. You can safely send your public key to another because your data cannot be decrypted without the other part of the key pair, your private key. Therefore, the owner must protect the private key that belongs to the public key in the digital certificate. All communications of the owner of a digital certificate can be deciphered, if the private key is known. Without the private key, a digital certificate cannot be misused.

Certification authorities and trust hierarchies:

A digital certificate is only as trustworthy as the certification authority (CA) that issued it.

As part of this trust, the policies under which certificates are issued should be understood. Each organization or user must determine which certification authorities can be accepted as trustworthy.

The Key Manager tool also allows organizations to create self-signed certificates, which can be useful for testing or in environments with a small number of users or machines.

As a user of a security service, you need to know its public key to obtain and validate any digital certificates. Also, simply receiving a digital certificate does not assure its authenticity. To verify its authenticity, you need the public key of the certification authority that issued that digital certificate. If you do not already hold an assured copy of the CA's public key, then you might need an additional digital certificate to obtain the CA's public key.

Certificate revocation lists

A digital certificate is expected to be used for its entire validity period. If needed, however, a certificate can be invalidated before its actual date of expiration.

Invalidating the certificate might be necessary, for example, if an employee leaves the company or if the certificate's private key has been compromised. To invalidate a certificate, you must notify the appropriate Certificate Authority (CA) of the circumstances. When a CA revokes a certificate, it adds the invalid certificate serial number to a Certificate Revocation List (CRL).

CRLs are signed data structures that are issued periodically and made available in a public repository. CRLs can be retrieved from HTTP or LDAP servers. Each CRL contains a current time stamp and a nextUpdate time stamp. Each revoked certificate in the list is identified by its certificate serial number.

When configuring an IKE tunnel and using digital certificates as your authentication method, you can confirm the certificate has not been revoked by selecting RSA Signature with CRL Checking. If CRL Checking is enabled, the list is located and checked during the negotiation process to establish the key management tunnel.

Note: To use this feature of IP Security, your system must be configured to use a SOCKS server (version 4 for HTTP servers), an LDAP server, or both. If you know which SOCKS or LDAP server you are using to obtain CRLs, you can make the necessary configuration selections by using Web-based System Manager. Select **CRL Configuration** from the Digital Certificates menu.

Uses for digital certificates in Internet applications

Internet applications that use public-key cryptography systems must use digital certificates to obtain the public keys.

There are many applications that use public-key cryptography, including the following ones:

Virtual Private Networks (VPN)

Virtual Private Networks, also called *secure tunnels*, can be set up between systems such as firewalls to enable protected connections between secure networks over unsecured communication links. All traffic destined to these networks is encrypted between the participating systems.

The protocols used in tunneling follow the IP Security and IKE standards, which allow for a secure, encrypted connection between a remote client (for example, an employee working from home) and a secure host or network.

Secure Sockets Layer (SSL)

SSL is a protocol that provides privacy and integrity for communications. It is used by Web servers for secure connections between Web servers and Web browsers, by the Lightweight Directory Access Protocol (LDAP) for secure connections between LDAP clients and LDAP servers, and by Host-on-Demand V.2 for connections between the client and the host system. SSL uses digital certificates for key exchange, server authentication, and, optionally, client authentication.

Secure Electronic Mail

Many electronic mail systems, using standards such as PEM or S/MIME for secure electronic mail, use digital certificates for digital signatures and for the exchange of keys to encrypt and decrypt mail messages.

Digital certificates and certificate requests

A *certificate request* must be created and sent to a CA to request a digital certificate.

A signed digital certificate contains fields for the owner’s distinguished name, the owner’s public key, the CA’s distinguished name and the CA’s signature. A self-signed digital certificate contains its owner’s distinguished name, public key, and signature.

The certificate request contains fields for the requestor’s distinguished name, public key, and signature. The CA verifies the requestor’s signature with the public key in the digital certificate to ensure that:

- The certificate request was not modified in transit between the requestor and the CA.
- The requestor possesses the corresponding private key for the public key that is in the certificate request.

The CA is also responsible for verifying to some level the identity of the requestor. Requirements for this verification can range from very little proof to absolute assurance of the owner’s identity.

Key Manager tool

The Key Manager tool manages digital certificates, and is located in the **gskkm.rte** file set on the expansion pack.

This section describes how to use Key Manager to do the following tasks:

- Creating a Key Database
- Adding a CA Root Digital Certificate
- Establishing Trust Settings
- Deleting a CA Root Digital Certificate
- Requesting a Digital Certificate
- Adding (Receiving) a New Digital Certificate
- Deleting a Digital Certificate
- Changing a Database Password
- Creating IKE Tunnels using Digital Certificates

To set up digital certificates and signature support, at minimum you must do tasks 1, 2, 3, 4, 6, and 7. Then, use Web-based System Manager to create an IKE tunnel and associate a policy with the tunnel that uses RSA Signature as the authentication method.

You can create and configure a key database from the Web-based System Manager VPN Overview window by selecting the Managing Digital Certificates option, or by using the certmgr command to open the Key Manager tool from the command line.

Creating a key database:

A key database enables VPN endpoints to connect using valid digital certificates. The key database (*.kdb) format is used with IP Security VPNs.

The following types of CA digital certificates are provided with Key Manager:

- RSA Secure Server Certification Authority
- Thawte Personal Premium Certification Authority
- Thawte Personal Freemail Certification Authority
- Thawte Personal Basic Certification Authority
- Thawte Personal Server Certification Authority
- Thawte Server Certification Authority
- Verisign Class 1 Public Primary Certification Authority
- Verisign Class 2 Public Primary Certification Authority
- Verisign Class 3 Public Primary Certification Authority
- Verisign Class 4 Public Primary Certification Authority

These signature digital certificates enable clients to attach to servers that have valid digital certificates from these signers. After you create a key database, you can use it as created to attach to a server that has a valid digital certificate from one of the signers.

To use a signature digital certificate that is not on this list, you must request it from the CA and add it to your key database. See “Adding a CA root digital certificate” on page 163.

To create a key database using the **certmgr** command, use the following procedure:

1. Start the Key Manager tool by typing:

   ```
   # certmgr
   ```

2. Select **New** from the Key Database File list.
3. Accept the default value, CMS key database file, for the **Key database type** field.
4. Enter the following file name in the **File Name** field:

   ```
   ikekey.kdb
   ```
5. Enter the following location of the database in the **Location** field:

   ```
   /etc/security
   ```

   Note: The key database must be named **ikekey.kbd** and it must be placed in the **/etc/security** directory. Otherwise, IP Security cannot function correctly.
6. Click **OK**. The **Password Prompt** screen is displayed.
7. Enter a password in the **Password** field, and enter it again in the **Confirm Password** field.
8. If you want to change the number of days until the password expires, enter the desired number of days in the **Set expiration time?** field. The default value for this field is 60 days. If you do not want the password to expire, clear the **Set expiration time?** field.
9. To save an encrypted version of the password in a stash file, select the **Stash the password to a file?** field and enter Yes.

   Note: You must stash the password to enable the use of digital certificates with IP Security.
10. Click **OK**. A confirmation screen displays, verifying that you have created a key database.
11. Click **OK** again and you return to the IBM Key Management screen. You can either perform other tasks or exit the tool.

Adding a CA root digital certificate:

After you have requested and received a root digital certificate from a CA, you can add it to your database.

Most root digital certificates are of the form *.arm, such as the following example:

```
cert.arm
```

To add a CA root digital certificate to a database, use the following procedure:

1. Unless you are already using Key Manager, start the tool by typing:

   ```
   # certmgr
   ```
2. From the main screen, select **Open** from the Key Database File list.
3. Highlight the key database file to which you want to add a CA root digital certificate and click **Open**.
4. Enter the password and click **OK**. When your password is accepted, you are returned to the IBM Key Management screen. The title bar now shows the name of the key database file you selected, indicating that the file is now open and ready to be worked with.
5. Select **Signer Certificates** from the **Personal/Signer Certificates** list.
6. Click **Add**.
7. Select a data type from the **Data type** list, such as:

   ```
   Base64-encoded ASCII data
   ```
8. Enter a certificate file name and location for the CA root digital certificate, or click **Browse** to select the name and location.
9. Click **OK**.
10. Enter a label for the CA root digital certificate, such as `Test CA Root Certificate`, and click **OK**. You are returned to the **Key Management** screen. The **Signer Certificates** field now shows the label of the CA root digital certificate you just added. You can either perform more tasks or exit the tool.

Establishing trust settings:

Installed CA certificates are set to trusted by default.

To change the trust setting, do the following steps:

1. Unless you are already using Key Manager, start the tool by typing:
   ```
   # certmgr
   ```
2. From the main screen, select **Open** from the **Key Database File** list.
3. Highlight the key database file in which you want to change the default digital certificate and click **Open**.
4. Enter the password and click **OK**. After your password is accepted, you are returned to the **IBM Key Management** screen. The title bar shows the name of the key database file you selected, indicating that the file is now open.
5. Select **Signer Certificates** from the **Personal/Signer Certificates** list.
6. Highlight the certificate you want to change and click **View/Edit**, or double-click on the entry. The **Key Information** screen is displayed for the certificate entry.
7. To make this certificate a trusted root certificate, select the check box next to **Set the certificate as a trusted root** and click **OK**. If the certificate is not trusted, clear the check box instead and click **OK**.
8. Click **OK** from the **Signer Certificates** screen. You are returned to the **IBM Key Management** screen. You can either perform other tasks or exit the tool.

Deleting a CA root digital certificate:

If you no longer want to use one of the CAs in your signature digital certificate list, you must delete the CA root digital certificate.

Note: Before deleting a CA root digital certificate, create a backup copy in case you later want to recreate the CA root.

To delete a CA root digital certificate from a database, use the following procedure:

1. Unless you are already using Key Manager, start the tool by typing:
   ```
   # certmgr
   ```
2. From the main screen, select **Open** from the **Key Database File** list.
3. Highlight the key database file from which you want to delete a CA root digital certificate and click **Open**.
4. Enter the password and click **OK**. After your password is accepted, you are returned to the **Key Management** screen. The title bar shows the name of the key database file you selected, indicating that the file is now open and ready to be edited.
5. Select **Signer Certificates** from the **Personal/Signer Certificates** list.
6. Highlight the certificate you want to delete and click **Delete**. The **Confirm** screen is displayed.
7. Click **Yes**. You are returned to the **IBM Key Management** screen. The label of the CA root digital certificate no longer appears in the **Signer Certificates** field. You can either perform other tasks or exit the tool.

Requesting a digital certificate:

To acquire a digital certificate, generate a request using Key Manager and submit the request to a CA. The request file you generate is in the PKCS#10 format. The CA then verifies your identity and sends you a digital certificate.

To request a digital certificate, use the following procedure:

1. Unless you are already using Key Manager, start the tool by typing:
   ```
   # certmgr
   ```

2. From the main screen, select **Open** from the **Key Database File** list.
3. Highlight the **/etc/security/ikekey.kdb** key database file from which you want to generate the request and click **Open**.
4. Enter the password and click **OK**. After your password is accepted, you are returned to the **IBM Key Management** screen. The title bar shows the name of the key database file you selected, indicating that the file is now open and ready to be edited.
5. Select **Personal Certificate Requests** from the **Personal/Signer Certificates** list (in AIX Version 4) or select **Create** → **New Certificate Request** (beginning in AIX 5.1).
6. Click **New**.
7. From the following screen, enter a Key Label for the self-signed digital certificate, such as:

   ```
   keytest
   ```
8. Enter a common name (the default is the host name) and organization, and then select a country. For the remaining fields, either accept the default values, or choose new values.
9. Define the subject alternate name. The optional fields associated with subject alternate are e-mail address, IP address, and DNS name. For a tunnel type of IP address, type the same IP address that is configured in the IKE tunnel into the IP address field. For a tunnel ID type of *user@FQDN*, complete the e-mail address field. For a tunnel ID type of FQDN, type a fully qualified domain name (for example, *hostname*.*companyname*.com) in the DNS name field.
10. At the bottom of the screen, enter a name for the file, such as:

    ```
    certreq.arm
    ```
11. Click **OK**. A confirmation screen is displayed, verifying that you have created a request for a new digital certificate.
12. Click **OK**. You are returned to the **IBM Key Management** screen. The **Personal Certificate Requests** field now shows the key label of the new digital certificate request (PKCS#10) created.
13. Send the file to a CA to request a new digital certificate. You can either perform other tasks or exit the tool.

Adding (Receiving) a new digital certificate:

After you receive a new digital certificate from a CA, you must add it to the key database from which you generated the request.

To add (receive) a new digital certificate, use the following procedure:

1. Unless you are already using Key Manager, start the tool by typing:

   ```
   # certmgr
   ```
2. From the main screen, select **Open** from the **Key Database File** list.
3. Highlight the key database file from which you generated the certificate request and click **Open**.
4. Enter the password and click **OK**. After your password is accepted, you are returned to the IBM Key Management screen. The title bar shows the name of the key database file you selected, indicating that the file is now open and ready to be edited.
5. Select **Personal Certificate Requests** from the **Personal/Signer Certificates** list.
6. Click **Receive** to add the newly received digital certificate to your database.
7. Select the data type of the new digital certificate from the **Data type** list. The default is **Base64-encoded ASCII data**.
8. Enter the certificate file name and location for the new digital certificate, or click **Browse** to select the name and location.
9. Click **OK**.
10. Enter a descriptive label for the new digital certificate, such as:

    ```
    VPN Branch Certificate
    ```

11. Click **OK**. You are returned to the **IBM Key Management** screen. The **Personal Certificates** field now shows the label of the new digital certificate you just added. You can either perform other tasks or exit the tool. If there is an error loading the certificate, check that the certificate file begins with the text `—-BEGIN CERTIFICATE—-` and ends with the text `—-END CERTIFICATE—-`.

    For example:

    ```
    -----BEGIN CERTIFICATE-----
    ajdkfjaldfwwwwwwwwwwadafdw
    kajf;kdsajkflasasfkjafdaff
    akdjf;ldasjkf;safdfdasfdas
    kaj;fdljk98dafdas43adfadfa
    -----END CERTIFICATE-----
    ```

    If the text does not match, edit the certificate file so that it starts and ends appropriately.

Deleting a digital certificate:

At times it will be necessary to delete a digital certificate.

Note: Before deleting a digital certificate, create a backup copy in case you later want to re-create it.

To delete a digital certificate from your database, use the following procedure:

1. Unless you are already using Key Manager, start the tool by typing:

   ```
   # certmgr
   ```
2. From the main screen, select **Open** from the **Key Database File** list.
3. Highlight the key database file from which you want to delete the digital certificate and click **Open**.
4. Enter the password and click **OK**. After your password is accepted, you are returned to the **IBM Key Management** screen. The title bar shows the name of the key database file you selected, indicating that the file is now open and ready to be edited.
5. Select **Personal Certificate Requests** from the **Personal/Signer Certificates** list.
6. Highlight the digital certificate you want to delete and click **Delete**. The **Confirm** screen is displayed.
7. Click **Yes**. You are returned to the **IBM Key Management** screen. The label of the digital certificate you just deleted no longer appears in the **Personal Certificates** field. You can either perform other tasks or exit the tool.

Changing a database password:

At times it will be necessary to change a database password.

To change the key database, use the following procedure:

1. Unless you are already using Key Manager, start the tool by typing:

   ```
   # certmgr
   ```
2. From the main screen, select **Change Password** from the **Key Database File** list.
3. Enter a new password in the **Password** field, and enter it again in the **Confirm Password** field.
4. If you want to change the number of days until the password expires, enter the desired number of days in the **Set expiration time?** field. The default value for this field is 60 days. If you do not want the password to expire, clear the **Set expiration time?** field.
5. To save an encrypted version of the password in a stash file, select the **Stash the password to a file?** field and enter `Yes`.

   Note: You must stash the password to enable the use of digital certificates with IP Security.
6. Click **OK**. A message in the status bar indicates that the request completed successfully.
7. Click **OK** again and you return to the **IBM Key Management** screen. You can either perform other tasks or exit the tool.

Creating IKE tunnels using digital certificates:

To create IKE tunnels that use digital certificates, you must use Web-based System Manager and the Key Manager tool.

To enable the use of digital certificates when defining the key management IKE tunnel policies, you must configure a transform that uses signature mode. Signature mode uses an RSA signature algorithm for authentication. IP Security provides the Web-based System Manager dialog ″Add/Change a Transform″ to allow you to select an authentication method of RSA Signature or RSA Signature with CRL Checking.

At least one endpoint of the tunnel must have a policy defined that uses a signature mode transform. You can also define other transforms using signature mode through Web-based System Manager.

The IKE key management tunnel types (the **Host Identity Type** field on the **Identification** tab) supported by IP Security are as follows:

- IP address
- Fully Qualified Domain Name (FQDN)
- *user@FQDN*
- X.500 Distinguished Name
- Key identifier

Use **Web-based System Manager** to select host-identity types on the **Key Management Tunnel Properties - Identification** tab. If you select **IP Address, FQDN**, or **user@FQDN**, you must enter values in Web-based System Manager and then provide these values to the CA. This information is used as the Subject Alternate Name in the personal digital certificate.

For example, if you choose a host identity type of X.500 Distinguished Name from the Web-based System Manager list on the **Identification** tab, and you enter the host identity as **/C=US/O=ABC/OU=SERV/CN=*name*.austin.ibm.com**, the following are the values that you must enter in Key Manager when creating a digital certificate request:

- Common name: ***name*.austin.ibm.com**
- Organization: **ABC**
- Organizational unit: **SERV**
- Country : **US**

The X.500 Distinguished Name entered is the name set up by your system or LDAP administrator. Entering an organizational unit value is optional. The CA then uses this information when creating the digital certificate.

For another example, if you choose a host identity type of **IP Address** from the list, and you enter the host identity as **10.10.10.1**, the following are the values you must enter in the digital certificate request:

- Common name: ***name*.austin.ibm.com**
- Organization: **ABC**
- Organizational unit: **SERV**
- Country : **US**
- Subject alternate IP address field: **10.10.10.1**

After you create the digital certificate request with this information, the CA uses this information to create the personal digital certificate.

When requesting a personal digital certificate, the CA needs the following information:

- You are requesting a X.509 certificate.

- The signature format is MD5 with RSA encryption.
- Whether you are specifying Subject Alternate Name. Alternate name types are:
  - IP address
  - Fully qualified domain name (FQDN)
  - *user@FQDN*

  The following subject alternate-name information is included in the certificate request file.
- Your planned key use (the digital signature bit must be selected).
- The Key Manager digital certificate request file (in PKCS#10 format).

For specific steps using the Key Manager tool to create a certificate request, see “Requesting a digital certificate” on page 164.

Before activating the IKE tunnel, you must add the personal digital certificate you received from the CA into the Key Manager database, **ikekey.kdb**. For more information, see “Adding (Receiving) a new digital certificate” on page 165.

IP Security supports the following types of personal digital certificates:

Subject DN
The Subject Distinguished Name must be in the following format and order:

`/C=US/O=ABC/OU=SERV/CN=`*`name`*`.austin.ibm.com`

The Key Manager tool allows only one **OU** value.

Subject DN and Subject Alternate Name as an IP address
The Subject Distinguished Name and Subject Alternate Name can be designated as an IP address, as shown in the following:

`/C=US/O=ABC/OU=SERV/CN=name.austin.ibm.com` and `10.10.10.1`

Subject DN and Subject Alternate Name as FQDN
The Subject Distinguished Name and Subject Alternate Name can be designated as a fully qualified domain name, as shown in the following:

`/C=US/O=ABC/OU=SERV/CN=name.austin.ibm.com` and `bell.austin.ibm.com`.

Subject DN and Subject Alternate Name as *user@FQDN*
The Subject Distinguished Name and Subject Alternate Name can be designated as a user address (*user_ID@fully_qualified_domain_name*), as shown in the following:

`/C=US/O=ABC/OU=SERV/CN=name.austin.ibm.com` and `name@austin.ibm.com`.

Subject DN and multiple Subject Alternate Names
The Subject Distinguished Name can be associated with multiple Subject Alternate Names, as shown in the following:

`/C=US/O=ABC/OU=SERV/CN=name.austin.ibm.com` and `bell.austin.ibm.com`, `10.10.10.1`, and `user@name.austin.ibm.com`.

Network address translation

IP Security can use devices whose addresses undergo network address translation (NAT).

NAT is widely used as part of firewall technology for Internet-connection sharing, and it is a standard feature on routers and edge devices. The IP Security protocol depends on identifying remote endpoints and their policy based on the remote IP address. When intermediate devices such as routers and firewalls translate a private address to a public address, the required authentication processing in IP Security might fail because the address in the IP packet has been modified after the authentication digest was calculated. With the new IP Security NAT support, devices that are configured behind a node that performs network address translation are able to establish an IP Security Tunnel. The IP Security code is able to detect when a remote address has been translated. Using the new IP Security implementation with support for NAT allows a VPN client to connect from home or on the road to the office through an internet connection with NAT enabled.

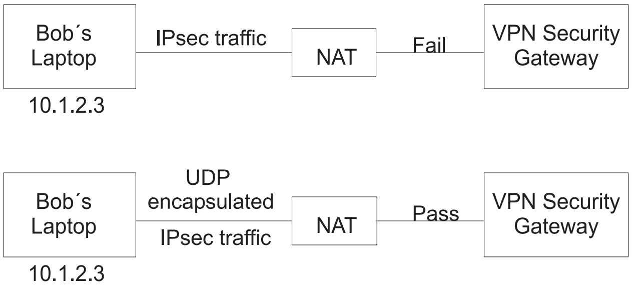


Figure 12. NAT-enabled IP Security

This diagram shows the difference between a NAT-enabled IP Security implementation, with UDP encapsulated traffic and an implementation that is not NAT-enabled.

Configuring IP security to work with NAT

In order to use NAT in IP Security, you must set the *ENABLE_IPSEC_NAT_TRAVERSAL* variable in the **/etc/isakmpd.conf** file. When this variable is set, filter rules are added to send and receive traffic on port 4500.

The following example shows the filter rules when the *ENABLE_IPSEC_NAT_TRAVERSAL* variable is set.

```
Dynamic rule 2:
Rule action         : permit
Source Address      : 0.0.0.0 (any)
Source Mask         : 0.0.0.0 (any)
Destination Address : 0.0.0.0 (any)
Destination Mask    : 0.0.0.0 (any)
Source Routing      : no
Protocol            : udp
Source Port         : 0 (any)
Destination Port    : 4500
Scope               : local
Direction           : inbound
Fragment control    : all packets
Tunnel ID number    : 0

Dynamic rule 3:
Rule action         : permit
Source Address      : 0.0.0.0 (any)
Source Mask         : 0.0.0.0 (any)
Destination Address : 0.0.0.0 (any)
Destination Mask    : 0.0.0.0 (any)
Source Routing      : no
Protocol            : udp
Source Port         : 4500
Destination Port    : 0 (any)
Scope               : local
Direction           : outbound
Fragment control    : all packets
Tunnel ID number    : 0
```

Setting the *ENABLE_IPSEC_NAT_TRAVERSAL* variable also adds some additional filter rules in the filter table. Special IPSEC NAT messages use UDP encapsulation and filter rules must be added to allow this traffic to flow. In addition, in phase 1 signature mode is required. If IP Address is used as the identifier in the certificate, it should contain the private ip address.

IP Security also needs to send NAT keep alive messages to maintain the mapping of the original IP Address and the NAT address. The interval is specified by the *NAT_KEEPALIVE_INTERVAL* variable in **/etc/isakmpd.conf** file. This variable specifies how frequently NAT keepalive packets are sent in seconds. If you do not specify a value for *NAT_KEEPALIVE_INTERVAL*, a default value of 20 seconds is used.

Limitations when using NAT exchanges

Endpoints behind NAT devices must protect their traffic using the ESP protocol.

ESP is the predominate header selected for IP Security, and will be usable for most customer applications. ESP includes hashing of the user data, but not of the IP Header. The integrity checking in the AH header incorporates the IP source and destination addresses in the keyed message integrity check. NAT or reverse NAT devices that make changes to the address fields invalidate the message integrity check. Therefore, if only the AH protocol is defined in the phase 2 policy for a tunnel, and NAT is detected in a phase 1 exchange, a Notify Payload saying `NO_PROPOSAL_CHOSEN` is sent.

Additionally, a connection using NAT must select tunnel mode so that the original IP address is encapsulated in the packet. Transport mode and addresses with NAT are not compatible. If a NAT is detected and only transport mode is proposed in phase 2, a Notify Payload saying `NO_PROPOSAL_CHOSEN` is sent.

Avoiding tunnel mode conflicts

Remote peers might negotiate entries that overlap in a gateway. This overlap causes a tunnel mode conflict.

The following figure shows a tunnel mode conflict.

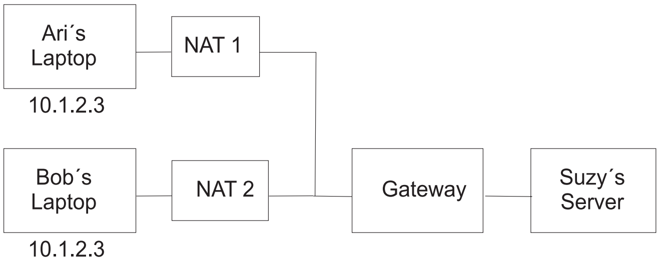


Figure 13. Tunnel Mode Conflict

The gateway has two possible Security Associations (SAs) for the 10.1.2.3 IP address. These duplicate remote addresses cause confusion over where to send packets coming from the server. When a tunnel is configured between Suzy's server and Ari's laptop, the IP address is used, and Suzy cannot configure a tunnel with Bob with the same IP address. To avoid a tunnel mode conflict, you should not define a tunnel with the same IP address. Because the remote address is not under the control of the remote user, other ID types should be used to identify the remote host such as fully qualified domain name or user at fully qualified domain name.

Configuring manual tunnels

The following procedures configure IP Security to use manual tunnels.

Setting up tunnels and filters

The process of setting up a tunnel is to define the tunnel on one end, import the definition on the other end, and activate the tunnel and filter rules on both ends. The tunnel is then ready to use.

To set up a manual tunnel, it is not necessary to separately configure the filter rules. As long as all traffic between two hosts goes through the tunnel, the necessary filter rules are automatically generated.

Information about the tunnel must be made to match on both sides if it is not explicitly supplied. For instance, the encryption and authentication algorithms specified for the source will be used for the destination if the destination values are not specified.

Creating a manual tunnel on the first host

You can configure a tunnel using the Web-based System Manager Network application, the SMIT**ips4_basic** fast path (for IP Version 4), the SMIT **ips6_basic** fast path (for IP version 6) or you can create the tunnel manually using the following procedure.

The following is a sample of the **gentun** command used to create a manual tunnel:

```
gentun -v 4 -t manual -s 5.5.5.19 -d 5.5.5.8 \
   -a HMAC_MD5 -e DES_CBC_8 -N 23567
```

You can use the **lstun -v 4** command to list the characteristics of the manual tunnel created by the previous example. The output looks similar to the following example:

```
Tunnel ID            : 1
IP Version           : IP Version 4
Source               : 5.5.5.19
Destination          : 5.5.5.8
Policy               : auth/encr
Tunnel Mode          : Tunnel
Send AH Algo         : HMAC_MD5
Send ESP Algo        : DES_CBC_8
Receive AH Algo      : HMAC_MD5
Receive ESP Algo     : DES_CBC_8
Source AH SPI        : 300
Source ESP SPI       : 300
Dest AH SPI          : 23576
Dest ESP SPI         : 23576
Tunnel Life Time     : 480
Status               : Inactive
Target               : -
Target Mask          : -
Replay               : No
New Header           : Yes
Snd ENC-MAC Algo     : -
Rcv ENC-MAC Algo     : -
```

To activate the tunnel, type the following code:

```
mktun -v 4 -t1
```

The filter rules associated with the tunnel are automatically generated.

To view the filter rules, use the **lsfilt -v 4** command. The output looks similar to the following example:

```
Rule 4:
Rule action            : permit
Source Address         : 5.5.5.19
Source Mask            : 255.255.255.255
Destination Address    : 5.5.5.8
Destination Mask       : 255.255.255.255
Source Routing         : yes
Protocol               : all
Source Port            : any 0
Destination Port       : any 0
Scope                  : both
Direction              : outbound
Logging control        : no
Fragment control       : all packets
Tunnel ID number       : 1
Interface              : all
Auto-Generated         : yes
```

```
Rule 5:
Rule action          : permit
Source Address       : 5.5.5.8
Source Mask          : 255.255.255.255
Destination Address  : 5.5.5.19
Destination Mask     : 255.255.255.255
Source Routing       : yes
Protocol             : all
Source Port          : any 0
Destination Port     : any 0
Scope                : both
Direction            : inbound
Logging control      : no
Fragment control     : all packets
Tunnel ID number     : 1
Interface            : all
Auto-Generated       : yes
```

To activate the filter rules, including the default filter rules, use the **mktun -v 4 -t 1** command.

To set up the other side (when it is another machine using this operating system), the tunnel definition can be exported on host A and then imported to host B.

The following command exports the tunnel definition into a file named **ipsec_tun_manu.exp** and any associated filter rules to the file **ipsec_fltr_rule.exp** in the directory indicated by the **-f** flag:

```
exptun -v 4 -t 1 -f /tmp
```

Creating a manual tunnel on the second host

To create the matching end of the tunnel, the export files are copied and imported into the remote machine.

Use the following command to create the matching end of the tunnel:

```
imptun -v 4 -t 1 -f /tmp
```

where

1 Is the tunnel to be imported

/tmp Is the directory where the import files reside

The tunnel number is generated by the system. You can obtain it from the output of the **gentun** command or by using the **lstun** command to list the tunnels and determine the correct tunnel number to import. If there is only one tunnel in the import file, or if all the tunnels are to be imported, the **-t** option is not needed.

If the remote machine is not running this operating system, the export file can be used as a reference for setting up the algorithm, keys, and security parameters index (SPI) values for the other end of the tunnel.

Export files from a firewall product can be imported to create tunnels. To do this, use the **-n** option when importing the file, as follows:

```
imptun -v 4 -f /tmp -n
```

Setting up filters

Filtering can be set up to be simple, using mostly autogenerated filter rules, or can be customized by defining very specific filter functions based on the properties of the IP packets.

Each line in a filter table is known as a *rule*. A collection of rules determine what packets are accepted in and out of the machine and how they are directed. Matches to filter rules on incoming packets are done by comparing the source address and SPI value to those listed in the filter table. Therefore, this pair must be unique. Filter rules can control many aspects of communications, including source and destination addresses and masks, protocol, port number, direction, fragment control, source routing, tunnel, and interface type.

The types of filter rules are as follows:

- "Static filter rules" are created in the filter table to be used for the general filtering of traffic or for associating with manual tunnels. They can be added, deleted, modified, and moved. An optional description text field can be added to identify a specific rule.
- "Autogenerated filter rules and user-specified filter rules" on page 176 (also called *autogenerated* filter rules) are a specific set of rules created for use of IKE tunnels. Both static and dynamic filter rules are created based on data management tunnel information and on data management tunnel negotiation.
- "Predefined filter rules" on page 177 are generic filter rules that cannot be modified, moved, or deleted, such as the `all traffic` rule, the `ah` rule, and the `esp` rule. They pertain to all traffic.

Associated with these filter rules are *Subnet masks*, which group IDs that are associated with a filter rule, and the host-firewall-host configuration option. The following sections describe the different types of filter rules and their associated features.

Static filter rules

Each static filter rule contains space-separated fields.

The following list provides the name of each field in a static filter rule followed by an example from rule 1 in parentheses:

- Rule_number (`1`)
- Action (`permit`)
- Source_addr (`0.0.0.0`)
- Source_mask (`0.0.0.0`)
- Dest_addr (`0.0.0.0`)
- Dest_mask (`0.0.0.0`)
- Source_routing (`no`)
- Protocol (`udp`)
- Src_prt_operator (`eq`)
- Src_prt_value (`4001`)
- Dst_prt_operator (`eq`)
- Dst_prt_value (`4001`)
- Scope (`both`)
- Direction (`both`)
- Logging (`no`)
- Fragment (`all packets`)
- Tunnel (`0`)
- Interface (`all`).

Example of static filter rules

```
1 permit 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 no udp eq 4001 eq 4001 both both no all
   packets 0 all

2 permit 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 no ah any 0 any 0 both both no all packets
   0 all

3 permit 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 no esp any 0 any 0 both both no all packets
   0 all
```

```
4 permit 10.0.0.1 255.255.255.255 10.0.0.2 255.255.255.255 no all any 0 any 0 both
   outbound no all packets 1 all outbound traffic

5 permit 10.0.0.2 255.255.255.255 10.0.0.1 255.255.255.255 no all any 0 any 0 both
   inbound no all packets 1 all


6 permit 10.0.0.1 255.255.255.255 10.0.0.3 255.255.255.255 no tcp lt 1024 eq 514 local
   outbound yes all packets 2 all


7 permit 10.0.0.3 255.255.255.255 10.0.0.1 255.255.255.255 no tcp/ack eq 514 lt 1024
   local inbound yes all packets 2 all


8 permit 10.0.0.1 255.255.255.255 10.0.0.3 255.255.255.255 no tcp/ack lt 1024 lt 1024
   local outbound yes all packets 2 all


9 permit 10.0.0.3 255.255.255.255 10.0.0.1 255.255.255.255 no tcp lt 1024 lt 1024 local
   inbound yes all packets 2 all


10 permit 10.0.0.1 255.255.255.255 10.0.0.4 255.255.255.255 no icmp any 0 any 0 local
   outbound yes all packets 3 all


11 permit 10.0.0.4 255.255.255.255 10.0.0.1 255.255.255.255 no icmp any 0 any 0 local
   inbound yes all packets 3 all


12 permit 10.0.0.1 255.255.255.255 10.0.0.5 255.255.255.255 no tcp gt 1023 eq 21 local
   outbound yes all packets 4 all


13 permit 10.0.0.5 255.255.255.255 10.0.0.1 255.255.255.255 no tcp/ack eq 21 gt 1023 local
   inbound yes all packets 4 all


14 permit 10.0.0.5 255.255.255.255 10.0.0.1 255.255.255.255 no tcp eq 20 gt 1023 local
   inbound yes all packets 4 all


15 permit 10.0.0.1 255.255.255.255 10.0.0.5 255.255.255.255 no tcp/ack gt 1023 eq 20 local
   outbound yes all packets 4 all


16 permit 10.0.0.1 255.255.255.255 10.0.0.5 255.255.255.255 no tcp gt 1023 gt 1023 local
   outbound yes all packets 4 all


17 permit 10.0.0.5 255.255.255.255 10.0.0.1 255.255.255.255 no tcp/ack gt 1023 gt 1023 local
   inbound yes all packets 4 all


18 permit 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0 no all any 0 any 0 both both yes all
   packets
```

Each rule in the previous example is described as follows:

Rule 1

For the **Session Key** daemon. This rule only appears in IP Version 4 filter tables. It uses port number 4001 to control packets for refreshing the session key. Rule 1 an example of how the port number can be used for a specific purpose.

Note: Do not modify this filter rule, except for logging purposes.

Rules 2 and 3

Allow processing of authentication headers (AH) and encapsulating security payload (ESP) headers.

Note: Do not modify Rules 2 and 3, except for logging purposes.

Rules 4 and 5

Set of autogenerated rules that filter traffic between addresses 10.0.0.1 and 10.0.0.2 through tunnel 1. Rule 4 is for outbound traffic, and rule 5 is for inbound traffic.

Note: Rule 4 has a user-defined description of *outbound traffic*.

Rules 6 through 9

Set of user-defined rules that filter outbound rsh, rcp, rdump, rrestore, and rdist services between addresses 10.0.0.1 and 10.0.0.3 through tunnel 2. In this example, logging is set to Yes, so that the administrator can monitor this type of traffic.

Rules 10 and 11

Set of user-defined rules that filter both inbound and outbound icmp services of any type between addresses 10.0.0.1 and 10.0.0.4 through tunnel 3.

Rules 12 through 17

User-defined filter rules that filter outbound file transfer protocol (FTP) service from 10.0.0.1 and 10.0.0.5 through tunnel 4.

Rule 18

Autogenerated rule always placed at the end of the table. In this example, it permits all packets that do not match the other filter rules. It can be set to deny all traffic not matching the other filter rules.

Each rule can be viewed separately (using **lsfilt**) to list each field with its value. For example:

```
Rule 1:
Rule action          : permit
Source Address       : 0.0.0.0
Source Mask          : 0.0.0.0
Destination Address  : 0.0.0.0
Destination Mask     : 0.0.0.0
Source Routing       : yes
Protocol             : udp
Source Port          : eq  4001
Destination Port     : eq  4001
Scope                : both
Direction            : both
Logging control      : no
Fragment control     : all packets
Tunnel ID number     : 0
Interface            : all
Auto-Generated       : yes
```

The following list contains all the parameters that can be specified in a filter rule:

-v IP Version: 4 or 6.

-a Action:

- **d** Deny
- **p** Permit

-s Source address. Can be an IP address or hostname.

-m Source subnet mask.

-d Destination address. Can be an IP address or hostname.

-M Destination subnet mask.

-g Source routing control: `y` or `n`.

-c Protocol. Values can be `udp`, `icmp`, `tcp`, `tcp/ack`, `ospf`, `pip`, `esp`, `ah` and `all`.

-o Source port or ICMP type operation.

-p Source port or ICMP type value.

-O Destination port or ICMP code operation.

-P Destination port or ICMP code value.

-r Routing:

- **r** Forwarded packets.
- **l** Local destined/originated packets.
- **b** Both.

-l Log control.

- **y** Include in log.
- **n** Do not include in log.

-f Fragmentation.

- **y** Applies to fragments headers, fragments, and non-fragments.
- **o** Applies only to fragments and fragment headers.
- **n** Applies only to non-fragments.
- **h** Applies only to non-fragments and fragment headers.

-t Tunnel ID.

-i Interface, such as `tr0` or `en0`.

For more information, see the **genfilt** and **chfilt** command descriptions.

Autogenerated filter rules and user-specified filter rules

Certain rules are autogenerated for the use of the IP Security filter and tunnel code.

Autogenerated rules include:

- Rules for the session key daemon that refresh the IP version 4 keys in IKE (AIX 4.3.3 and later)
- Rules for the processing of AH and ESP packets.

Filter rules are also autogenerated when defining tunnels. For manual tunnels, autogenerated rules specify the source and destination addresses and the mask values, as well as the tunnel ID. All traffic between those addresses will flow through the tunnel.

For IKE tunnels, autogenerated filter rules determine protocol and port numbers during IKE negotiation. The IKE filter rules are kept in a separate table, which is searched after the static filter rules and before the autogenerated rules. IKE filter rules are inserted in a default position within the static filter table, but they can be moved by the user.

Autogenerated rules permit all traffic over the tunnel. User-defined rules can place restrictions on certain types of traffic. Place these user-defined rules before the autogenerated rules, because IP Security uses the first rule it finds that applies to the packet. The following is an example of user-defined filter rules that filter traffic based on ICMP operation.

```
1 permit 10.0.0.1 255.255.255.255 10.0.0.4 255.255.255.255 no icmp any 8 any 0
   local outbound no all packets 3 all
2 permit 10.0.0.4 255.255.255.255 10.0.0.1 255.255.255.255 no icmp any 0 any 0 local
   inbound no all packets 3 all
3 permit 10.0.0.4 255.255.255.255 10.0.0.1 255.255.255.255 no icmp any 8 any 0 local
   inbound no all packets 3 all
4 permit 10.0.0.1 255.255.255.255 10.0.0.4 255.255.255.255 no icmp any 0 any 0 local
   outbound no all packets 3 all
```

To simplify the configuration of a single tunnel, filter rules are autogenerated when tunnels are defined. This function can be suppressed by specifying the **-g** flag in the **gentun**. You can find a sample filter file with **genfilt** commands to generate filter rules for different TCP/IP services in **/usr/samples/ipsec/filter.sample**.

Predefined filter rules

Several predefined filter rules are autogenerated with certain events.

When the `ipsec_v4` or `ipsec_v6` device is loaded, a predefined rule is inserted into the filter table and then activated. By default, this predefined rule is to permit all packets, but it is user-configurable and you can set it to deny all packets.

Note: When configuring remotely, ensure that the deny rule is not enabled before the configuration is complete, to prevent your session from getting locked out of the machine. The situation can be avoided either by setting the default action to permit or by configuring a tunnel to the remote machine before activating IP Security.

Both IP Version 4 and IP Version 6 filter tables have a predefined rule. Either may be independently changed to deny all. This will keep traffic from passing unless that traffic is specifically defined by additional filter rules. The only other option to change on the predefined rules is **chfilt** with the **-l** option, which allows packets matching that rule to be logged.

To support IKE tunnels, a dynamic filter rule is placed in the IP Version 4 filter table. This is the position at which dynamic filter rules are inserted into the filter table. This position can be controlled by the user by moving its position up and down the filter table. After the tunnel manager daemon and **isakmpd** daemon are initialized to allow IKE tunnels to be negotiated, rules are automatically created in the dynamic filter table to handle IKE messages as well as AH and ESP packets.

Subnet masks

Subnet masks are used to group a set of IDs that are associated with a filter rule. The mask value is ANDed with the ID in the filter rules and compared to the ID specified in the packet.

For example, a filter rule with a source IP address of `10.10.10.4` and a subnet mask of `255.255.255.255` specified that an exact match must occur of the decimal IP address, as shown in the following:

	Binary	Decimal
Source IP address	`1010.1010.1010.0100`	`10.10.10.4`
Subnet mask	`11111111.11111111.11111111.11111111`	`255.255.255.255`

A `10.10.10.x` subnet is specified as `11111111.11111111.11111111.0` or `255.255.255.0`. An incoming address would have the subnet mask applied to it, then the combination would be compared to the ID in the filter rule. For example, an address of `10.10.10.100` becomes `10.10.10.0` after the subnet mask is applied, which matches the filter rule.

A subnet mask of `255.255.255.240` allows any value for the last four bits in the address.

Host-firewall-host configuration

The host-firewall-host configuration option for tunnels allows you to create a tunnel between your host and a firewall, then automatically generate the necessary filter rules for correct communication between your host and a host behind the firewall.

The autogenerated filter rules permit all rules between the two non-firewall hosts over the tunnel specified. The default rules—for user datagram protocol (UDP), Authentication Headers (AH), and Encapsulating Security Payload (ESP) headers—should already handle the host to firewall communication. The firewall will have to be configured appropriately to complete the setup. You should use the export file from the tunnel you created to enter the SPI values and keys that the firewall needs.

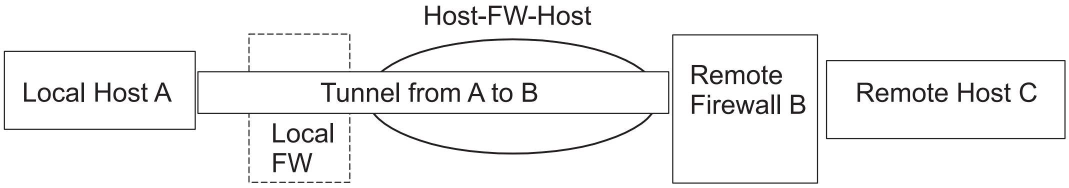


Figure 14. Host-Firewall-Host

This illustration shows a Host-Firewall-Host configuration. Host A has a tunnel running through a local firewall and out to the internet. Then it goes to Remote Firewall B, and then on to Remote Host C.

Logging facilities

This section describes the configuration and format of system logs relating to IP Security.

As hosts communicate with each other, the transferred packets may be logged to the system log daemon, **syslogd**. Other important messages about IP Security also display. An administrator may choose to monitor this logging information for traffic analysis and debugging assistance. The following are the steps for setting up the logging facilities.

1. Edit the **/etc/syslog.conf** file to add the following entry:

   ```
   local4.debug var/adm/ipsec.log
   ```

   Use the `local4` facility to record traffic and IP Security events. Standard operating system priority levels apply. You should set the priority level of `debug` until traffic through IP Security tunnels and filters show stability and proper movement.

   Note: The logging of filter events can create significant activity at the IP Security host and can consume large amounts of storage.
2. Save the **/etc/syslog.conf file**.
3. Go to the directory you specified for the log file and create an empty file with the same name. In the case above, you would change to **/var/adm** directory and issue the command:

   ```
   touch ipsec.log
   ```
4. Issue a **refresh** command to the **syslogd** subsystem:

   ```
   refresh -s syslogd
   ```
5. If you are using IKE tunnels, ensure the **/etc/isakmpd.conf** file specifies the desired **isakmpd** logging level. (See “Internet Protocol security problem diagnosis” on page 181 for more information on IKE logging.)
6. While creating filter rules for your host, if you would like packets matching a specific rule to be logged, set the *-l* parameter for the rule to **Y** (Yes) using the **genfilt** or the **chfilt** commands.
7. Turn on packet logging and start the **ipsec_logd** daemon using the command:

   ```
   mkfilt -g start
   ```

   You can stop packet logging by issuing the following command:

   ```
   mkfilt -g stop
   ```

The following sample log file contains traffic entries and other IP Security log entries:

```
1. Aug 27 08:08:40 host1 : Filter logging daemon ipsec_logd (level 2.20)
   initialized at 08:08:40 on 08/27/97A
2. Aug 27 08:08:46 host1 : mkfilt: Status of packet logging set to Start
   at 08:08:46 on 08/27/97
3. Aug 27 08:08:47 host1 : mktun: Manual tunnel 2 for IPv4, 9.3.97.244, 9.3.97.130
```

```
   activated.
4. Aug 27 08:08:47 host1 : mkfilt: #:1 permit 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0
   udp eq  4001 eq  4001  both both l=n f=y t=0 e= a=
5. Aug 27 08:08:47 host1 : mkfilt: #:2 permit 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0
   ah any 0 any 0  both both l=n f=y t=0 e= a=
6. Aug 27 08:08:47 host1 : mkfilt: #:3 permit 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0
   esp any 0 any 0  both both l=n f=y t=0 e= a=
7. Aug 27 08:08:47 host1 : mkfilt: #:4 permit 10.0.0.1 255.255.255.255 10.0.0.2
   255.255.255.255 icmp any 0 any 0  local outbound l=y f=y t=1 e= a=
8. Aug 27 08:08:47 host1 : mkfilt: #:4 permit 10.0.0.2 255.255.255.255 10.0.0.1
   255.255.255.255 icmp any 0 any 0  local inbound l=y f=y t=1 e= a=
9. Aug 27 08:08:47 host1 : mkfilt: #:6 permit 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0
    all any 0 any 0  both both l=y f=y t=0 e= a=
10. Aug 27 08:08:47 host1 : mkfilt: Filter support (level 1.00) initialized at
    08:08:47 on 08/27/97
11. Aug 27 08:08:48 host1 : #:6 R:p  o:10.0.0.1 s:10.0.0.1 d:10.0.0.20 p:udp
    sp:3327 dp:53 r:l a:n f:n T:0 e:n l:67
12. Aug 27 08:08:48 host1 : #:6 R:p  i:10.0.0.1 s:10.0.0.20 d:10.0.0.1 p:udp
    sp:53 dp:3327 r:l a:n f:n T:0 e:n l:133
13. Aug 27 08:08:48 host1 : #:6 R:p  i:10.0.0.1 s:10.0.0.15 d:10.0.0.1 p:tcp
    sp:4649 dp:23 r:l a:n f:n T:0 e:n l:43
14. Aug 27 08:08:48 host1 : #:6 R:p  o:10.0.0.1 s:10.0.0.1 d:10.0.0.15 p:tcp
    sp:23 dp:4649 r:l a:n f:n T:0 e:n l:41
15. Aug 27 08:08:48 host1 : #:6 R:p  i:10.0.0.1 s:10.0.0.15 d:10.0.0.1 p:tcp
    sp:4649 dp:23 r:l a:n f:n T:0 e:n l:40
16. Aug 27 08:08:51 host1 : #:4 R:p  o:10.0.0.1 s:10.0.0.1 d:10.0.0.2 p:icmp
    t:8 c:0 r:l a:n f:n T:1 e:n l:84
17. Aug 27 08:08:51 host1 : #:5 R:p  i:10.0.0.1 s:10.0.0.2 d:10.0.0.1 p:icmp
    t:0 c:0 r:l a:n f:n T:1 e:n l:84
18. Aug 27 08:08:52 host1 : #:4 R:p  o:10.0.0.1 s:10.0.0.1 d:10.0.0.2 p:icmp
    t:8 c:0 r:l a:n f:n T:1 e:n l:84
19. Aug 27 08:08:52 host1 : #:5 R:p  i:10.0.0.1 s:10.0.0.2 d:10.0.0.1 p:icmp
    t:0 c:0 r:l a:n f:n T:1 e:n l:84
20. Aug 27 08:32:27 host1 : Filter logging daemon terminating at 08:32:27 on
    08/27/97l
```

The following paragraphs explain the log entries.

1 Filter logging daemon activated.

2 Filter packet logging set to on with the **mkfilt -g start** command.

3 Tunnel activation, showing tunnel ID, source address, destination address, and time stamp.

4-9 Filters have been activated. Logging shows all loaded filter rules.

10 Message showing activation of filters.

11-12 These entries show a DNS lookup for a host.

13-15 These entries show a partial Telnet connection (the other entries have been removed from this example for space reasons).

16-19 These entries show two pings.

20 Filter logging daemon shutting down.

The following example shows two hosts negotiating a phase 1 and a phase 2 tunnel from the initiating host's point of view. (The **isakmpd** logging level has been specified as **isakmp_events**.)

```
1. Dec  6 14:34:42 host1 Tunnel Manager: 0: TM is processing a
    Connection_request_msg
 2. Dec  6 14:34:42 host1 Tunnel Manager: 1: Creating new P1 tunnel object (tid)
 3. Dec  6 14:34:42 host1 isakmpd: 192.168.100.103 >>> 192.168.100.104 ( SA PROPOSAL
    TRANSFORM  )
 4. Dec  6 14:34:42 host1 isakmpd: ::ffff:192.168.100.103 <<< 192.168.100.104 ( SA
    PROPOSAL TRANSFORM  )
 5. Dec  6 14:34:42 host1 isakmpd: Phase I SA Negotiated
 6. Dec  6 14:34:42 host1 isakmpd: 192.168.100.103 >>> 192.168.100.104 ( KE NONCE  )
 7. Dec  6 14:34:42 host1 isakmpd: ::ffff:192.168.100.103 <<< 192.168.100.104 ( KE
    NONCE  )
 8. Dec  6 14:34:42 host1 isakmpd: Encrypting the following msg to send: ( ID HASH
     )
 9. Dec  6 14:34:42 host1 isakmpd: 192.168.100.103 >>> 192.168.100.104 ( Encrypted
    Payloads )
10. Dec  6 14:34:42 host1 isakmpd: ::ffff:192.168.100.103 <<< 192.168.100.104 (
    Encrypted Payloads )
11. Dec  6 14:34:42 host1 Tunnel Manager: 1: TM is processing a P1_sa_created_msg
    (tid)
12. Dec  6 14:34:42 host1 Tunnel Manager: 1:   Received good P1 SA, updating P1
    tunnel (tid)
13. Dec  6 14:34:42 host1 Tunnel Manager: 0: Checking to see if any P2 tunnels need
    to start
14. Dec  6 14:34:42 host1 isakmpd: Decrypted the following received msg: ( ID HASH
     )
15. Dec  6 14:34:42 host1 isakmpd:  Phase I Done !!!
16. Dec  6 14:34:42 host1 isakmpd: Phase I negotiation authenticated
17. Dec  6 14:34:44 host1 Tunnel Manager: 0: TM is processing a
    Connection_request_msg
18. Dec  6 14:34:44 host1 Tunnel Manager: 0: Received a connection object for an
    active P1 tunnel
19. Dec  6 14:34:44 host1 Tunnel Manager: 1: Created blank P2 tunnel (tid)
20. Dec  6 14:34:44 host1 Tunnel Manager: 0: Checking to see if any P2 tunnels need
    to start
21. Dec  6 14:34:44 host1 Tunnel Manager: 1: Starting negotiations for P2 (P2 tid)
22. Dec  6 14:34:45 host1 isakmpd: Encrypting the following msg to send: ( HASH SA
    PROPOSAL TRANSFORM NONCE ID ID  )
23. Dec  6 14:34:45 host1 isakmpd: 192.168.100.103 >>> 192.168.100.104 ( Encrypted
    Payloads )
24. Dec  6 14:34:45 host1 isakmpd: ::ffff:192.168.100.103 <<< 192.168.100.104 (
    Encrypted Payloads )
25. Dec  6 14:34:45 host1 isakmpd: Decrypted the following received msg: ( HASH SA
    PROPOSAL TRANSFORM NONCE ID ID  )
26. Dec  6 14:34:45 host1 isakmpd: Encrypting the following msg to send: ( HASH  )
27. Dec  6 14:34:45 host1 isakmpd: 192.168.100.103 >>> 192.168.100.104 ( Encrypted
    Payloads )
28. Dec  6 14:34:45 host1 isakmpd: Phase II SA Negotiated
29. Dec  6 14:34:45 host1 isakmpd: PhaseII negotiation complete.
30. Dec  6 14:34:45 host1 Tunnel Manager: 0: TM is processing a P2_sa_created_msg
31. Dec  6 14:34:45 host1 Tunnel Manager: 1: received p2_sa_created for an existing
    tunnel as initiator (tid)
32. Dec  6 14:34:45 host1 Tunnel Manager: 1: Filter::AddFilterRules: Created filter
    rules for tunnel
33. Dec  6 14:34:45 host1 Tunnel Manager: 0: TM is processing a List_tunnels_msg
```

The following paragraphs explain the log entries.

1-2 The **ike cmd=activate phase=1** command initiates a connection.

3-10 The **isakmpd** daemon negotiates a phase 1 tunnel.

11-12 The Tunnel Manager receives a valid phase 1 security association from the responder.

13 The Tunnel Manager checks whether **ike cmd=activate** has a phase 2 value for more work. It does not.

14-16 The **isakmpd** daemon finishes the phase 1 negotiation.

17-21 The **ike cmd=activate phase=2** command initiates a phase 2 tunnel.

22-29 The **isakmpd** daemon negotiates a phase 2 tunnel.

30-31 The Tunnel Manager receives a valid phase 2 security association from responder.

32 The Tunnel Manager writes the dynamic filter rules.

33 The **ike cmd=list** command views the IKE tunnels.

Labels in field entries

This section describes the fields displayed in log entries.

The fields in the log entries are abbreviated to reduce DASD space requirements:

The rule number that caused this packet to be logged.

R Rule Type

- **p** Permit
- **d** Deny

i/o Direction the packet was traveling when it was intercepted by the filter support code. Identifies IP address of the adapter associated with the packet:

- For inbound (i) packets, this is the adapter that the packet arrived on.
- For outbound (o) packets, this is the adapter that the IP layer has determined should handle the transmission of the packet.

s Specifies the IP address of the sender of the packet (extracted from the IP header).

d Specifies the IP address of the intended recipient of the packet (extracted from the IP header).

p Specifies the high-level protocol that was used to create the message in the data portion of the packet. May be a number or name, for example: `udp`, `icmp`, `tcp`, `tcp/ack`, `ospf`, `pip`, `esp`, `ah`, or `all`.

sp/t Specifies the protocol port number associated with the sender of the packet (extracted from the TCP/UDP header). When the protocol is ICMP or OSPF, this field is replaced with **t**, which specifies the IP type.

dp/c Specifies the protocol port number associated with the intended recipient of the packet (extracted from the TCP/UDP header). When the protocol is ICMP, this field is replaced with **c**, which specifies the IP code.

- Specifies that no information is available

r Indicates whether the packet had any local affiliation.

- **f** Forwarded packets
- **l** Local packets
- **o** Outgoing
- **b** Both

l Specifies the length of a particular packet in bytes.

f Identifies if the packet is a fragment.

T Indicates the tunnel ID.

i Specifies what interface the packet came in on.

Internet Protocol security problem diagnosis

This section includes some hints and tips that might assist you when you encounter a problem.

Set up logging when IPSec is first configured. Logs are very useful in determining what occurs with the filters and tunnels. (For detailed log information, see “Logging facilities” on page 178.)

Troubleshooting manual tunnel errors

This section provides the descriptions of several possible tunnel errors, along with their solutions.

Error: Issuing **mktun** command results in the following error:

```
insert_tun_man4(): write failed : The requested resource is busy.
```

Problem: The tunnel you requested to activate is already active or you have colliding SPI values.

To fix: Issue the **rmtun** command to deactivate, then issue the mktun command to activate. Check to see if the SPI values for the failing tunnel match any other active tunnel. Each tunnel should have its own unique SPI values.

Error: Issuing **mktun** command results in the following error:

```
Device ipsec_v4 is in Defined status.
Tunnel activation for IP Version 4 not performed.
```

Problem: You have not made the IP Security device available.

To fix: Issue the following command:

```
mkdev -l ipsec -t 4
```

You might have to change **-t** option to 6 if you are getting the same error for IP Version 6 tunnel activation. The devices must be in available state. To check the IP Security device state, issue the following command:

```
lsdev -Cc ipsec
```

Error: Issuing a **gentun** command results in the following error:

```
Invalid Source IP address
```

Problem: You have not entered a valid IP address for the source address.

To fix: For IP Version 4 tunnels, check to see that you have entered an available IP Version 4 address for the local machine. You cannot use host names for the source when generating tunnels, you might only use host names for the destination.

For IP Version 6 tunnels, check to see that you entered an available IP Version 6 address. If you type `netstat -in` and no IP Version 6 addresses exist, run **/usr/sbin/autoconf6** (interface) for a link local autogenerated address (using MAC address) or use the **ifconfig** command to manually assign an address.

Error: Issuing a **gentun** command results in the following error:

```
Invalid Source IP address
```

Problem: You have not entered a valid IP address for the source address.

To fix: For IP Version 4 tunnels, check to see that you have entered an available IP Version 4 address for the local machine. You cannot use host names for the source when generating tunnels, you may only use host names for the destination.

For IP Version 6 tunnels, check to see that you entered an available IP Version 6 address. If you type `netstat -in` and no IP Version 6 addresses exist, run **/usr/sbin/autoconf6** (interface) for a link local auto-generated address (using MAC address) or use **ifconfig** to manually assign an address.

Error: Issuing **mktun** command results in the following error:

```
insert_tun_man4(): write failed : A system call received a parameter that is not valid.
```

Problem: Tunnel generation occurred with invalid ESP and AH combination or without the use of the new header format when necessary.

To fix: Check to see which authentication algorithms are in use by the particular tunnel in question. Remember that the HMAC_MD5 and HMAC_SHA algorithms require the new header format. The new header format can be changed using the SMIT fast path **ips4_basic** or the *-z* parameter with the **chtun** command. Also, remember that DES_CBC_4 cannot be used with the new header format.

Error: Starting IP Security from Web-based System Manager results in a `Failure` message.

Problem: The IP Security daemons are not running.

To fix: View which daemons are running by entering the **ps -ef** command. The following daemons are associated with IP Security:

- **tmd**
- **isakmpd**
- **cpsd**

The **cpsd** daemon is active only if the digital certificate code is installed (the fileset named **gskit.rte** or **gskkm.rte**) and you have configured the Key Manager tool to contain digital certificates.

If the daemons are not active, stop IP Security using Web-based System Manager and then restart it, which automatically starts the appropriate daemons.

Error: Trying to use IP Security results in the following error:

```
The installed bos.crypto is back level and must be updated.
```

Problem: The **bos.net.ipsec.*** files have been updated to a newer version, but the corresponding **bos.crypto.*** files have not.

To fix: Update the **bos.crypto.*** files to the version that corresponds with the updated **bos.net.ipsec.*** files.

Troubleshooting Internet Key Exchange tunnel errors

The following sections describe errors that can occur when using Internet Key Exchange (IKE) tunnels.

Internet Key Exchange tunnel process flow:

This section describes the process flow for the internet key exchange tunnel.

The IKE tunnels are set up by the communication of the **ike** command or the Web-based System Manager VPN panels with the following daemons:

Table 10. Daemons used by IKE tunnels.

tmd	Tunnel Manager daemon
isakmpd	IKE daemon
cpsd	Certificate proxy daemon

For IKE tunnels to be correctly set up, the **tmd** and **isakmpd** daemons must be running. If IP Security is set to start at reboot, these daemons start automatically. Otherwise, they must be started using Web-based System Manager.

The Tunnel Manager gives requests to the **isakmpd** command to start a tunnel. If the tunnel already exists or is not valid (for instance, has an invalid remote address), it reports an error. If negotiation has started, it may take some time, depending on network latency, for the negotiation to complete. The **ike cmd=list** command can list the state of the tunnel to determine if the negotiation was successful. Also, the Tunnel Manager logs events to **syslog** to the levels of debug, event, and information, which can be used to monitor the progress of the negotiation.

The sequence is as follows:

1. Use Web-based System Manager or the **ike** command to initiate a tunnel.
2. The **tmd** daemon gives the **isakmpd** daemon a connection request for key management (phase 1).
3. The **isakmpd** daemon responds with `SA created` or an error message.

4. The **tmd** daemon gives the **isakmpd** daemon a connection request for a data management tunnel (phase 2).
5. The **isakmpd** daemon responds with `SA created` or an error message.
6. Tunnel parameters are inserted into the kernel tunnel cache.
7. Filter rules are added to the kernel dynamic filter table.

When the machine is acting as a responder, the **isakmpd** daemon notifies the Tunnel Manager **tmd** daemon that a tunnel has been negotiated successfully and a new tunnel is inserted into the kernel. In such cases, the process starts with step 3 and continues until step 7, without the **tmd** daemon issuing connection requests.

Internet Key-Exchange logging:

The **isakmpd**, **tmd** and **cpsd** daemons log events to the SYSLOG facility.

For the **isakmpd** daemon, you enable logging using the **ike cmd=log** command. Next, you set the logging level in the **/etc/isakmpd.conf** configuration file. Use the **log_level** parameter to set the level of logging. Based on the amount of information that you want to log, set the level to none, errors, isakmp_events, or information.

For example, to specify that you want to log protocol information and implementation information, specify the following parameter:

```
log_level=INFORMATION
```

Tip: In versions earlier than AIX 5.1, the **isakmpd** daemon logs data to a separate file, which is also specified in **/etc/isakmpd.conf** file.

For example, to specify that you want to log protocol and implementation information, specify the following parameter:

The **isakmpd** daemon starts one of two processes: it sends a proposal, or it evaluates a proposal. If the proposal is accepted, a security association is created and the tunnel is set up. If the proposal is not accepted or the connection ends before the negotiation completes, the **isakmpd** daemon indicates an error. The entries in the SYSLOG facility from **tmd** indicate whether the negotiation succeeded. A failure caused by a certificate that was not valid is logged to the SYSLOG facility. To determine the exact cause of a failed negotiation, review the data in the logging file that is specified in **/etc/syslog.conf**.

The SYSLOG facility adds a prefix to each line of the log, noting the date and time, the machine, and the program. The following example uses `googly` as the machine name and `isakmpd` as the program name:

```
Nov 20 09:53:50 googly isakmpd: ISAKMP_MSG_HEADER
Nov 20 09:53:50 googly isakmpd:  Icookie : 0xef06a77488f25315, Rcookie :0x0000000000000000
Nov 20 09:53:51 googly isakmpd:  Next Payload : 1(SA), Maj Ver : 1, Min Ver : 0
Nov 20 09:53:51 googly isakmpd:  Xchg Type : 2 (ID protected), Flag= 0, Encr : No,COMMIT : No
Nov 20 09:53:51 googly isakmpd:  Msg ID  : 0x00000000
```

To improve clarity, use the **grep** command to extract log lines of interest (such as all **isakmpd** logging) and the **cut** command to remove the prefix from each line. The **isakmpd** log examples in the rest of this section have been tailored in a similar way.

The /etc/isakmdp.conf file

Use the **/etc/isakmdp.conf** file configure these items:

- The log configuration of the IKE daemons.
- The manner in which the **isakmpd** command processes a proposed negotiation (whether the **isakmpd** daemon can accept a main-mode negotiation from an unknown peer).

- The manner in which the certificate revocation list (CRL) processes the following data:
  - The SOCKS4 server information
  - The LDAP server information
  - Whether the Hypertext Transfer Protocol (HTTP) server or the LDAP server is queried first, when both servers are configured.

The /etc/isakmdp.conf file configuration options

Using the **/etc/isakmdp.conf** file, you can set the following options:

Log configuration

Determine the amount of information that you want to log. Then set the level. The IKE daemons use this option to specify the level of logging.

Syntax: `none | error | isakmp_events | information`

where the level has the following meaning:

none No logging. This is the default.

error Log protocol errors or appliation programming interface (API) errors.

isakmp_events
Log IKE protocol events or errors. Use this level when debugging a problem.

information
Log protocol information and implementation information.

Unrecognized IP address negotiation

You can set this option to YES or NO. When you set this option to YES, the local IKE database must contain an IP address for both phase-1 tunnel endpoints. You must specify YES for the host to accept an incoming main-mode tunnel. The IP address can be the primary ID or an optional IP address that is associated with some other ID type.

Set this option to NO to accept an incoming main-mode connection. When you set the option to NO, the host might accept the connection even when the IKE database does not specify IP addresses for the phase 1 endpoints. However, in order for the host to accept the connection, you must use certificate-based authentication. This allows a host with a dynamically assigned IP address to initiate a main mode tunnel to the machine.

If you do not specify this parameter, the default is NO.

Syntax: MAIN_MODE_REQUIRES_IP= YES | NO

SOCKS4 server configuration

The `SOCKS4_PORTNUM` option is optional. If you do not specify it, the default SOCKS-server port value of 1080 is used. The port value is used when the SOCKS server communicates with the HTTP server.

Syntax: *mnemonic* = *value*

where *mneumonic* and *value* can be the following values:

`SOCKS4_SERVER=` specifies the server name

`SOCKS4_PORTNUM=` specifies the SOCKS-server port number

`SOCKS4_USERID=` user ID

LDAP server configuration

Syntax: *mnemonic* = *value*

where *mnemonic* and *value* can be the following values:

`LDAP_SERVER=` specifies the LDAP server name

`LDAP_VERSION=` the version of the LDAP server (can be 2 or 3)

`LDAP_SERVERPORT=` the LDAP-server port number

`LDAP_SEARCHTIME=`client-search timeout value

CRL Fetch Order

This option defines whether the HTTP or LDAP server is queried first, when both servers are configured. The `CRL_FETCH_ORDER` option is optional. The default fetch order is HTTP first, then LDAP, depending on whether both HTTP and LDAP servers are configured.

Syntax: CRL_FETCH_ORDER= *protocol#*, *protocol#*

where *protocol#* can be `HTTP` or `LDAP`.

Parse payload logging function:

The security association (SA) between two end points is established by exchanging IKE messages. The Parse Payload function parses the messages in a human-readable format.

Parse payload logging can be enabled by editing the **/etc/isakmpd.conf** file. The logging entry in the **/etc/isakmpd.conf** file looks similar to the following:

```
information
```

The type of IKE payloads that Parse Payload logs depends on the content of the IKE message. Examples include SA Payload, Key Exchange Payload, Certificate Request Payload, Certificate Payload, and Signature Payload. The following is an example of a Parse Payload log in which an ISAKMP_MSG_HEADER is followed by five payloads:

```
ISAKMP_MSG_HEADER
        Icookie : 0x9e539a6fd4540990, Rcookie : 0x0000000000000000
        Next Payload : 1(SA), Maj Ver : 1, Min Ver : 0
        Xchg Type : 4 (Aggressive), Flag= 0, Encr : No,COMMIT : No
        Msg ID  : 0x00000000
        len     : 0x10e(270)
SA Payload:
        Next Payload : 4(Key Exchange), Payload len : 0x34(52)
        DOI          : 0x1(INTERNET)
        bitmask      : 1(SIT_IDENTITY_ONLY
   Proposal Payload:
        Next Payload : 0(NONE), Payload len : 0x28(40)
        Proposal # : 0x1(1), Protocol-ID : 1(ISAKMP)
        SPI size : 0x0(0), # of Trans : 0x1(1)
   Transform Payload:
        Next Payload : 0(NONE), Payload len : 0x20(32)
        Trans # : 0x1(1), Trans.ID : 1(KEY_IKE)
        Attr : 1(Encr.Alg     ), len=0x2(2)
        Value=0x1(1),(DES-cbc)
        Attr : 2(Hash Alg     ), len=0x2(2)
        Value=0x1(1),(MD5)
        Attr : 3(Auth Method  ), len=0x2(2)
        Value=0x3(3),(RSA Signature)
        Attr : 4(Group Desc   ), len=0x2(2)
        Value=0x1(1),(default 768-bit MODP group)
        Attr : 11(Life Type    ), len=0x2(2)
        Value=0x1(1),(seconds)
        Attr : 12(Life Duration), len=0x2(2)
        Value=0x7080(28800)
Key Payload:
        Next Payload : 10(Nonce), Payload len : 0x64(100)

        Key Data :
        33 17 68 10 91 1f ea da 38 a0 22 2d 84 a3 5d 5d
        a0 e1 1f 42 c2 10 aa 8d 9d 14 0f 58 3e c4 ec a3
        9f 13 62 aa 27 d8 e5 52 8d 5c c3 cf d5 45 1a 79
        8a 59 97 1f 3b 1c 08 3e 2a 55 9b 3c 50 cc 82 2c
        d9 8b 39 d1 cb 39 c2 a4 05 8d 2d a1 98 74 7d 95
        ab d3 5a 39 7d 67 5b a6 2e 37 d3 07 e6 98 1a 6b

Nonce Payload:
        Next Payload : 5(ID), Payload len : 0xc(12)

        Nonce Data:
        6d 21 73 1d dc 60 49 93
ID Payload:
        Next Payload : 7(Cert.Req), Payload len : 0x49(73)
        ID type      : 9(DER_DN), Protocol : 0, Port = 0x0(0)
Certificate Request Payload:
        Next Payload : 0(NONE), Payload len : 0x5(5)
        Certificate Encoding Type: 4(X.509 Certificate - Signature)
```

Within each payload, a **Next Payload** field points to the payload following the current payload. If the current payload is the last one in the IKE message, the **Next Payload** field has the value of zero (None).

Each Payload in the example has information pertaining to the negotiations that are going on. For example, the SA payload has the Proposal and Transform Payloads, which in turn show the encryption algorithm, authentication mode, hash algorithm, SA life type, and SA duration that the initiator is proposing to the responder.

Also, the SA Payload consists of one or more Proposal Payloads and one or more Transform Payloads. The **Next Payload** field for Proposal Payload has a value of either 0 if it is the only Proposal Payload or a value of 2 if it is followed by one more Proposal Payloads. Similarly, the **Next Payload** field for a Transform Payload has a value of 0 if it is the only Transform Payload, or a value of 3 if it is followed by one more Transform Payloads, as shown in the following example:

```
ISAKMP_MSG_HEADER
        Icookie : 0xa764fab442b463c6, Rcookie : 0x0000000000000000
        Next Payload : 1(SA), Maj Ver : 1, Min Ver : 0
        Xchg Type : 2 (ID protected), Flag= 0, Encr : No,COMMIT : No
        Msg ID  : 0x00000000
        len     : 0x70(112)
SA Payload:
        Next Payload : 0(NONE), Payload len : 0x54(84)
        DOI          : 0x1(INTERNET)
        bitmask      : 1(SIT_IDENTITY_ONLY
   Proposal Payload:
        Next Payload : 0(NONE), Payload len : 0x48(72)
        Proposal # : 0x1(1), Protocol-ID : 1(ISAKMP)
        SPI size : 0x0(0), # of Trans : 0x2(2)
   Transform Payload:
        Next Payload : 3(Transform), Payload len : 0x20(32)
        Trans # : 0x1(1), Trans.ID : 1(KEY_IKE)
        Attr : 1(Encr.Alg     ), len=0x2(2)
        Value=0x5(5),(3DES-cbc)
        Attr : 2(Hash Alg     ), len=0x2(2)
        Value=0x1(1),(MD5)
        Attr : 3(Auth Method  ), len=0x2(2)
        Value=0x1(1),(Pre-shared Key)
        Attr : 4(Group Desc   ), len=0x2(2)
        Value=0x1(1),(default 768-bit MODP group)
        Attr : 11(Life Type    ), len=0x2(2)
        Value=0x1(1),(seconds)
        Attr : 12(Life Duration), len=0x2(2)
        Value=0x7080(28800)
   Transform Payload:
        Next Payload : 0(NONE), Payload len : 0x20(32)
        Trans # : 0x2(2), Trans.ID : 1(KEY_IKE)
        Attr : 1(Encr.Alg     ), len=0x2(2)
        Value=0x1(1),(DES-cbc)
        Attr : 2(Hash Alg     ), len=0x2(2)
        Value=0x1(1),(MD5)
        Attr : 3(Auth Method  ), len=0x2(2)
        Value=0x1(1),(Pre-shared Key)
        Attr : 4(Group Desc   ), len=0x2(2)
        Value=0x1(1),(default 768-bit MODP group)
        Attr : 11(Life Type    ), len=0x2(2)
        Value=0x1(1),(seconds)
        Attr : 12(Life Duration), len=0x2(2)
        Value=0x7080(28800)
```

The IKE message header of a Parse Payload log shows the exchange type (Main Mode or Aggressive Mode), the length of the entire message, the message identifier, and so on.

The Certificate Request Payload requests a certificate from the responder. The responder sends the certificate in a separate message. The following example shows the Certificate Payload and Signature Payload that are sent to a peer as a part of an SA negotiation. The certificate data and the signature data are printed in hex format.

```
ISAKMP_MSG_HEADER
        Icookie : 0x9e539a6fd4540990, Rcookie : 0xc7e0a8d937a8f13e
        Next Payload : 6(Certificate), Maj Ver : 1, Min Ver : 0
        Xchg Type : 4 (Aggressive), Flag= 0, Encr : No,COMMIT : No
        Msg ID  : 0x00000000
        len     : 0x2cd(717)
Certificate Payload:

        Next Payload : 9(Signature), Payload len : 0x22d(557)
        Certificate Encoding Type: 4(X.509 Certificate - Signature)
        Certificate: (len 0x227(551) in bytes
        82 02 24 30 82 01 8d a0 03 02 01 02 02 05 05 8e
        fb 3e ce 30 0d 06 09 2a 86 48 86 f7 0d 01 01 04
        05 00 30 5c 31 0b 30 09 06 03 55 04 06 13 02 46
        49 31 24 30 22 06 03 55 04 0a 13 1b 53 53 48 20
        43 6f 6d 6d 75 6e 69 63 61 74 69 6f 6e 73 20 53
        65 63 75 72 69 74 79 31 11 30 0f 06 03 55 04 0b
        13 08 57 65 62 20 74 65 73 74 31 14 30 12 06 03
        55 04 03 13 0b 54 65 73 74 20 52 53 41 20 43 41
        30 1e 17 0d 39 39 30 39 32 31 30 30 30 30 30 30
        5a 17 0d 39 39 31 30 32 31 32 33 35 39 35 39 5a
        30 3f 31 0b 30 09 06 03 55 04 06 13 02 55 53 31
        10 30 0e 06 03 55 04 0a 13 07 49 42 4d 2f 41 49
        58 31 1e 30 1c 06 03 55 04 03 13 15 62 61 72 6e
        65 79 2e 61 75 73 74 69 6e 2e 69 62 6d 2e 63 6f
        6d 30 81 9f 30 0d 06 09 2a 86 48 86 f7 0d 01 01
        01 05 00 03 81 8d 00 30 81 89 02 81 81 00 b2 ef
        48 16 86 04 7e ed ba 4c 14 d7 83 cb 18 40 0a 3f
        55 e9 ad 8f 0f be c5 b6 6d 19 ec de 9b f5 01 a6
        b9 dd 64 52 34 ad 3d cd 0d 8e 82 6a 85 a3 a8 1c
        37 e4 00 59 ce aa 62 24 b5 a2 ea 8d 82 a3 0c 6f
        b4 07 ad 8a 02 3b 19 92 51 88 fb 2c 44 29 da 72
        41 ef 35 72 79 d3 e9 67 02 b2 71 fa 1b 78 13 be
        f3 05 6d 10 4a c7 d5 fc fe f4 c0 b8 b8 fb 23 70
        a6 4e 16 5f d4 b1 9e 21 18 82 64 6d 17 3b 02 03
        01 00 01 a3 0f 30 0d 30 0b 06 03 55 1d 0f 04 04
        03 02 07 80 30 0d 06 09 2a 86 48 86 f7 0d 01 01
        04 05 00 03 81 81 00 75 a4 ee 9c 3a 18 f2 de 5d
        67 d4 1c e4 04 b4 e5 b8 5e 9f 56 e4 ea f0 76 4a
        d0 e4 ee 20 42 3f 20 19 d4 25 57 25 70 0a ea 41
        81 3b 0b 50 79 b5 fd 1e b6 0f bc 2f 3f 73 7d dd
        90 d4 08 17 85 d6 da e7 c5 a4 d6 9a 2e 8a e8 51
        7e 59 68 21 55 4c 96 4d 5a 70 7a 50 c1 68 b0 cf
        5f 1f 85 d0 12 a4 c2 d3 97 bf a5 42 59 37 be fe
        9e 75 23 84 19 14 28 ae c4 c0 63 22 89 47 b1 b6
        f4 c7 5d 79 9d ca d0
Signature Payload:
        Next Payload : 0(NONE), Payload len : 0x84(132)
```

```
Signature: len 0x80(128) in bytes
9d 1b 0d 90 be aa dc 43 95 ba 65 09 b9 00 6d 67
b4 ca a2 85 0f 15 9e 3e 8d 5f e1 f0 43 98 69 d8
5c b6 9c e2 a5 64 f4 ef 0b 31 c3 cb 48 7c d8 30
e3 a2 87 f4 7c 9d 20 49 b2 39 00 fa 8e bf d9 b0
7d b4 8c 4e 19 3a b8 70 90 88 2c cf 89 69 5d 07
f0 5a 81 58 2e 15 40 37 b7 c8 d6 8c 5c e2 50 c3
4d 19 7e e0 e7 c7 c2 93 42 89 46 6b 5f f8 8b 7d
5b cb 07 ea 36 e5 82 9d 70 79 9a fe bd 6c 86 36
```

Digital certificate and signature mode problems:

This section lists possible digital certificate and signature mode problems you can encounter, and provides corresponding solutions.

Error: The **cpsd** (Certificate Proxy Server daemon) does not start. An entry similar to the following appears in the log file:

```
Sep 21 16:02:00 ripple CPS[19950]: Init():LoadCaCerts() failed, rc=-12
```

Problem: The certificate database has not opened or has not been found.

To Fix: Ensure that the Key Manager certificate databases are present in **/etc/security**. The following files make up the database: **ikekey.crl**, **ikekey.kdb**, **ikekey.rdb**, **ikekey.sth**.

If only the **ikekey.sth** file is missing, the `stash password` option was not selected when the Key Manager database was created. The password must be stashed to enable using digital certificates with IP Security. (See Creating a Key Database for more information.)

Error: Key Manager gives the following error when receiving a certificate:

```
Invalid Base64-encoded data was found
```

Problem: Superfluous data has been found in the certificate file or else data was lost or corrupted.

To Fix: The ’DER’ Encoded Certificate should be contained within the following strings (shown below). No other characters should precede or follow other than the BEGIN and END CERTIFICATE strings.

```
-----BEGIN CERTIFICATE-----
MIICMTCCAZqgAwIBAgIFFKZtANowDQYJKoZIhvcNAQEFBQAwXDELMAkGA1UEBhMC
RkkxJDAiBgNVBAoTG1NTSCBDb21tdW5pY2F0aW9ucyBTZWN1cml0eTERMA8GA1UE
CxMIV2ViIHRlc3QxFDASBgNVBAMTC1Rlc3QgUlNBIENBMB4XDTk5MDkyMTAwMDAw
MFoXDTk5MTAyMTIzNTk1OVowOzELMAkGA1UEBhMCVVMxDDAKBgNVBAoTA0lCTTEe
MBwGA1UEAxMVcmlwcGxlLmF1c3Rpbi5pYm0uY29tMIGfMA0GCSqGSIb3DQEBAQUA
A4GNADCBiQKBgQC5EZqo6n7tZrpAL6X4L7mf4yXQSm+m/NsJLhp6afbFpPvXgYWC
wq4pvOtvxgum+FHrE0gysNjbKkE4Y6ixC9PGGAKHnhM3vrmvFjnl1G6KtyEz58Lz
BWW39QS6NJ1LqqP1nT+y3+Xzvfv8Eonqzno8mglCWMX09SguLmWoU1PcZQIDAQAB
oyAwHjALBgNVHQ8EBAMCBaAwDwYDVR0RBAgwBocECQNhhzANBgkqhkiG9w0BAQUF
AOBgQA6bgp4Zay34/fyAlyCkNNAYJRrN3Vc4NHN7IGjUziN6jK5UyB5zL37FERW
hT9ArPLzK7yEZs+MDNvB0bosyGWEDYPZr7EZHhYcoBP4/cd0V5rBFmA8Y2gUthPi
Ioxpi4+KZGHYyLqTrm+8Is/DVJaQmCGRPynHK35xjT6WuQtiYg==
-----END CERTIFICATE-----
```

The following options can help you diagnose and solve this problem.

- If data was lost or corrupted, recreate the Certificate
- Use an ASN.1 parser (available on the Internet World Wide Web) to check whether the certificate is valid by parsing the certificate successfully.

Error: Key Manager gives the following error when receiving a personal certificate:

```
No request key was found for the certificate
```

Problem: A Personal Certificate Request does not exist for the personal certificate being received.

To Fix: Create the Personal Certificate Request again and request a new certificate.

Error: Web-based System Manager gives the following error when you configure an IKE tunnel:

```
Error 171 in the Key Management (Phase 1) Tunnel operation:
PUT_IRL_FAILED
```

Problem: One cause for this error is that the host identity type, which is configured on the IKE dialog (Identification tab), is invalid. This can happen when the host identity type selected from the pull-down list does not logically match the type entered in the `Host Identity` field. For example, if you select a host identity type of X500 Distinguished Name, you must to enter a properly formatted distinguished name in the **Host Identity** field.

To Fix: Ensure the distinguished name you enter is correct for the type selected in the host identity pull-down list.

Error: An IKE negotiation fails and an entry similar to the following appears in the log file:

```
inet_cert_service::channelOpen():clientInitIPC():error,rc =2
(No such file or directory)
```

Problem: The **cpsd** is not running or has stopped.

To Fix: Start IP Security using Web-based System Manager. This action also starts the appropriate daemons.

Error: An IKE negotiation fails and an entry similar to the following appears in the log file:

```
CertRepo::GetCertObj:  DN Does Not Match:  ("/C=US/O=IBM/CN=ripple.austin.ibm.com")
```

Problem: The X.500 Distinguished Name (DN) entered while defining the IKE tunnel does not match the X.500 DN in the personal certificate.

To Fix: Change the IKE tunnel definition in Web-based System Manager to match the distinguished name in the certificate.

Error: While defining IKE tunnels in Web-based System Manager, the Digital certificate check box is disabled under the **Authentication Method** tab.

Problem: The policy associated with this tunnel does not use RSA signature mode authentication.

To Fix: Change the transform of the associated policy to use the RSA signature authentication method. For example, you can choose IBM_low_CertSig as a key management policy when defining a IKE tunnel.

Tracing facilities

Tracing is a debugging facility for tracing kernel events. Traces can be used to get more specific information about events or errors occurring in the kernel filter and tunnel code.

The SMIT IP Security trace facility is available through the Advanced IP Security Configuration menu. The information captured by this trace facility includes information on Error, Filter, Filter Information, Tunnel, Tunnel Information, Capsulation/Decapsulation, Capsulation Information, Crypto, and Crypto Information. By design, the error trace hook provides the most critical information. The info trace hook can generate critical information and may have an impact on system performance. This tracing provides clues as to what a problem might be. Tracing information is also required when speaking with a service technician. To access the tracing facility, use the SMIT fast path **smit ips4_tracing** (for IP Version 4) or **smit ips6_tracing** (for IP Version 6).

ipsecstat command

You can use the **ipsecstat** command to list the status of IP Security devices, IP Security crypto algorithms, and statistics of IP Security packets.

Issuing the **ipsecstat** command will generate the following sample report, which shows that the IP Security devices are in the available state, that there are three authentication algorithms installed, three encryption algorithms installed, and that there is a current report of packet activity. This information could be useful to you in determining where a problem exists if you are troubleshooting your IP Security traffic.

```
IP Security Devices:
ipsec_v4 Available
ipsec_v6 Available

Authentication Algorithm:
HMAC_MD5 -- Hashed MAC MD5 Authentication Module
HMAC_SHA -- Hashed MAC SHA Hash Authentication Module
KEYED_MD5 -- Keyed MD5 Hash Authentication Module

Encryption Algorithm:
CDMF -- CDMF Encryption Module
DES_CBC_4 -- DES CBC 4 Encryption Module
DES_CBC_8 -- DES CBC 8 Encryption Module
3DES_CBC -- Triple DES CBC Encryption Module

IP Security Statistics -
Total incoming packets:  1106
Incoming AH packets:326
Incoming ESP packets:  326
Srcrte packets allowed:  0
Total outgoing packets:844
Outgoing AH packets:527
Outgoing ESP packets:  527
Total incoming packets dropped:  12
  Filter denies on input:  12
  AH did not compute: 0
  ESP did not compute:0
  AH replay violation:0
  ESP replay violation:  0
Total outgoing packets dropped:0
  Filter denies on input:0
Tunnel cache entries added: 7
Tunnel cache entries expired:  0
Tunnel cache entries deleted:  6
```

Note: Beginning with AIX 4.3.2, there is no need to use CDMF because DES is now available worldwide. Reconfigure any tunnels that use CDMF to use DES or Triple DES.

IP security reference

List of commands

This section provides a list of commands.

ike cmd=activate	Starts an Internet Key Exchange (IKE) negotiation (AIX 4.3.3 and later).
ike cmd=remove	Deactivates IKE tunnels (AIX 4.3.3 and later)
ike cmd=list	Lists IKE tunnels (AIX 4.3.3 and later)
ikedb	Provides the interface to the IKE tunnel database(AIX 5.1 and later)
gentun	Creates a tunnel definition
mktun	Activates tunnel definition(s)
chtun	Changes a tunnel definition
rmtun	Removes a tunnel definition
lstun	Lists tunnel definition(s)
exptun	Exports tunnel definition(s)
imptun	Imports tunnel definition(s)
genfilt	Creates a filter definition
mkfilt	Activates filter definition(s)
mvfilt	Moves a filter rule
chfilt	Changes a filter definition
rmfilt	Removes a filter definition
lsfilt	Lists filter definition(s)
expfilt	Exports filter definition(s)

impfilt	Imports filter definition(s)
ipsec_convert	Lists status of IP security
ipsecstat	Lists status of IP security
ipsectrcbuf	Lists the contents of IP security tracing buffer
unloadipsec	Unloads a crypto module

List of methods

This section provides a list of methods.

defipsec	Defines an instance of IP Security for IP Version 4 or IP Version 6
cfgipsec	Configures and loads **ipsec_v4** or **ipsec_v6**
ucfgipsec	Unconfigures **ipsec_v4** or **ipsec_v6**

IP security migration

This section explains how to migrate your IKE tunnels, filters and pre-shared keys from AIX 4.3 to AIX 5.2.

Migrating IKE tunnels:

This section explains how to migrate IKE tunnels.

To migrate your tunnels, complete the following steps on a system running AIX 4.3:

1. Run the **bos.net.ipsec.keymgt.pre_rm.sh** script. When you run this script, the following files are created in the **/tmp** directory:
   a. **p2proposal.bos.net.ipsec.keymgt**
   b. **p1proposal.bos.net.ipsec.keymgt**
   c. **p1policy.bos.net.ipsec.keymgt**
   d. **p2policy.bos.net.ipsec.keymgt**
   e. **p1tunnel.bos.net.ipsec.keymgt**
   f. **p2tunnel.bos.net.ipsec.keymgt**

   Attention: Run this script only once. If you update the database and run the script again, you will lose all of the files, and you can not retrieve them. Read the script in "The bos.net.ipsec.keymgt.pre_rm.sh script" on page 193 before you migrate your tunnels.
2. Save the files created by the script and the **/tmp/lpplevel** file to some external media, such as a CD or floppy disk.

Migrating pre-shared keys:

This section provides the steps to update the pre-shared key format.

The IKE tunnel pre-shared key database is also corrupted during migration. To update the pre-shared key format, complete the following steps on the system that has been migrated to AIX 5.2:

1. Save the output of the **ikedb -g** command by running the following command:
   ```
   ikedb -g > out.keys
   ```
2. Edit the **out.keys** file to replace FORMAT=ASCII with FORMAT=HEX for the pre-shared key format.
3. Input the XML file by running the following command:
   ```
   ikedb -pF out.keys
   ```

Migrating filters:

This section explains how to migrate filters.

1. Export the filter rules files to the **/tmp** directory using SMIT by completing the following steps:
   a. Run the **smitty ipsec4** command.
   b. Select **Advanced IP Security Configuration—>Configure IP Security Filter Rules—>Export IP Security filter rules**.
   c. Enter **/tmp** for the directory name.
   d. Under the Filter Rules option press F4 and select **all** from the list.
   e. Press enter to save the filter rules in the **/tmp/ipsec_fltr_rule.exp** file on the external media.

   Complete this process for all of the systems you are migrating from AIX 4.3 to AIX 5.2.
2. Copy the six tunnel files created by the script, the **/tmp/lpplevel** file, and the **/tmp/ipsec_fltr_rule.exp** file to the **/tmp** directory on the migrated system.
3. Run the **bos.net.ipsec.keymgt.post_i.sh** script to repopulate the tunnel configurations into the database.
4. Run the **ikedb -g** command to verify that the tunnels are in the database.

   Note: If you do not see the tunnel information in the database, run the script again, but rename all the ***.loaded** files in **/tmp** directory to their original names.

On a system that has been migrated to AIX 5.2, the filter database is corrupted after migration. If you run the **lsfilt** command on the migrated system, you will get the following error:

```
Cannot get ipv4 default filter rule
```

To update the filter database, complete the following steps:

1. Replace the **ipsec_filter** file and the **ipsec_filter.vc** file in the **/etc/security** directory with the uncorrupted files from a newly migrated system running AIX 5.2. If you do not have these files, you can request them from IBM Service.
2. Import the filter rules files to the **/tmp** directory using SMIT by completing the following steps:
   a. Run the **smitty ipsec4** command.
   b. Select **Advanced IP Security Configuration—>Configure IP Security Filter Rules—>Import IP Security filter rules**.
   c. Enter **/tmp** for the directory name.
   d. Under the **Filter Rules** option press **F4** and select **all** from the list.
   e. Press Enter to recreate the filter rules. You can list the filter rules through SMIT or with the **lsfilt** command.

Migration scripts:

The bos.net.ipsec.keymgt.pre_rm.sh script:

The **bos.net.ipsec.keymgt.pre_rm.sh** script saves the contents of the tunnel database on a system running AIX 4.3.

```
#!/usr/bin/ksh
keymgt_installed=`lslpp -Lqc bos.net.ipsec.keymgt 2>/dev/null | awk -F: '{print $6}' | head -1`

if [ ! "$keymgt_installed" ]
then
  exit 0
fi


#  Copy the database to a save directory in case changes fail
if [ -d /etc/ipsec/inet/DB ]
then
  cp -R /etc/ipsec/inet/DB /etc/ipsec/inet/DB.sav || exit $?
fi
```

```
#  Remember the level you are migrating from
VRM=$(LANG=C lslpp -Lqc bos.net.ipsec.keymgt 2>/dev/null | awk -F: '{print $3}' | \
awk -F. '{print $1"."$2"."$3}')
VR=${VRM%.*}
echo $VRM > /tmp/lpplevel

IKEDB=$(which ikedb) || IKEDB=/usr/sbin/ikedb

XMLFILE=/tmp/full_ike_database.bos.net.ipsec.keymgt
PSKXMLFILE=/tmp/psk_ike_database.bos.net.ipsec.keymgt

#  See if ikedb exists.
if [ -f $IKEDB ]
then

  # If either of the ikedb calls below fails, that's OK.  Just remove the
  # resulting file (which may contain garbage) and continue.  The post_i
  # script will simply not import the file if it doesn't exist, which will
  # mean part or all of the IKE database is lost, but this is preferable
  # to exiting the script with an error code, which causes the entire
  # migration to fail.

  $IKEDB -g > $XMLFILE
  if [ $? -ne 0 ]
  then
    rm -f $XMLFILE  || exit $?
  fi

  if [[ $VR = "5.1" ]]; then
    # This is a special case.  The 5.1 version of ikedb is the only
    # one that does not include preshared keys in the full database
    # output.  So we have to retrieve those separately.
    $IKEDB -g -t IKEPresharedKey > $PSKXMLFILE
    if [ $? -ne 0 ]
    then
     rm -f $PSKXMLFILE  || exit $?
    fi
  fi

#  Make sure ikegui command is installed
elif [ -f /usr/sbin/ikegui ]
then

  #  Get database information and save to /tmp
  /usr/sbin/ikegui 0 1 0 0 > /tmp/p1proposal.bos.net.ipsec.keymgt 2>/dev/null
  RC=$?
  if [[ $RC -ne 0 ]]
  then
    rm -f /tmp/p1proposal.bos.net.ipsec.keymgt || exit $?
  fi

  /usr/sbin/ikegui 0 1 1 0 > /tmp/p1policy.bos.net.ipsec.keymgt 2>/dev/null
  RC=$?
  if [[ $RC -ne 0 ]]
  then
    rm -f /tmp/p1policy.bos.net.ipsec.keymgt || exit $?
  fi

  /usr/sbin/ikegui 0 2 0 0 > /tmp/p2proposal.bos.net.ipsec.keymgt 2>/dev/null
  RC=$?
  if [[ $RC -ne 0 ]]
  then
    rm -f /tmp/p2proposal.bos.net.ipsec.keymgt || exit $?
  fi

  /usr/sbin/ikegui 0 2 1 0 > /tmp/p2policy.bos.net.ipsec.keymgt 2>/dev/null
  RC=$?
  if [[ $RC -ne 0 ]]
  then
    rm -f /tmp/p2policy.bos.net.ipsec.keymgt || exit $?
  fi

  /usr/sbin/ikegui 0 1 2 0 > /tmp/p1tunnel.bos.net.ipsec.keymgt 2>/dev/null
  RC=$?
  if [[ $RC -ne 0 ]]
  then
    rm -f /tmp/p1tunnel.bos.net.ipsec.keymgt || exit $?
  fi

  /usr/sbin/ikegui 0 2 2 0 > /tmp/p2tunnel.bos.net.ipsec.keymgt 2>/dev/null
  RC=$?
  if [[ $RC -ne 0 ]]
  then
    rm -f /tmp/p2tunnel.bos.net.ipsec.keymgt || exit $?
  fi

fi
```

The bos.net.ipsec.keymgt.post_i.sh script:

The **bos.net.ipsec.keymgt.post_i.sh** script loads the contents of the tunnel database on to a migrated system running AIX 5.2.

```
#!/usr/bin/ksh

function PrintDot {
    echo "echo \c"
    echo "\".\c"
    echo "\\\c\c"
    echo "\"\c"
    echo
}

function P1PropRestore {
    while :
    do
        read NAME
        read MODE
        if [[ $? = 0 ]]; then
            echo "ikegui 1 1 0 $NAME $MODE \c"
            MORE=1
            while [[ $MORE = 1 ]];
            do
                read AUTH
                read HASH
                read ENCRYPT
                read GROUP
                read TIME
                read SIZE
                read MORE
                echo "$AUTH $HASH $ENCRYPT $GROUP $TIME $SIZE $MORE \c"
            done
            echo " > /dev/null 2>&1"
            PrintDot
        else
            return 0
        fi
    done
}

function P2PropRestore {
    while :
    do
```

```
        read NAME
        FIRST=yes
        MORE=1
        while [[ $MORE = 1 ]];
        do
            read PROT
            if [[ $? = 0 ]]; then
                read AH_AUTH
                read ESP_ENCR
                read ESP_AUTH
                read ENCAP
                read TIME
                read SIZE
                read MORE
                if [[ $FIRST = "yes" ]]; then
                    echo "ikegui 1 2 0 $NAME $MODE \c"
                fi
                echo "$PROT $AH_AUTH $ESP_ENCR $ESP_AUTH $ENCAP $TIME $SIZE $MORE \c"
                FIRST=no
            else
                return 0
            fi
        done
        echo " > /dev/null 2>&1"
        PrintDot
    done
}

function P1PolRestore {
    while :
    do
        read NAME
        read ROLE
        if [[ $? = 0 ]]; then
            read TIME
            read SIZE
            read OVERLAP
            read TTIME
            read TSIZE
            read MIN
            read MAX
            read PROPOSAL
            echo "ikegui 1 1 1 $NAME $ROLE $OVERLAP $TTIME $TSIZE $MIN $MAX 1 0 0 $PROPOSAL > \
/dev/null 2>&1"
            PrintDot
        else
            return 0
        fi
    done
}

function P2PolRestore {
    while :
    do
        read NAME
        read ROLE
        if [[ $? = 0 ]]; then
            read IPFS
            read RPFS
            read TIME
            read SIZE
            read OVERLAP
            read TTIME
            read TSIZE
            read MIN
            read MAX
            echo "ikegui 1 2 1 $NAME $ROLE $IPFS $RPFS $OVERLAP $TTIME $TSIZE $MIN $MAX 1 0 0 \c"
            MORE=1
            while [[ $MORE = 1 ]];
            do
                read PROPOSAL
                read MORE
                echo "$PROPOSAL $MORE \c"
                FIRST=no
            done
        else
            return 0
        fi
        echo " > /dev/null 2>&1"
        PrintDot
    done
}

function P1TunRestore {
    while :
    do
        read TUNID
        read NAME
        if [[ $? = 0 ]]; then
            read LID_TYPE
            read LID
            if [[ $LPPLEVEL = "4.3.3" ]]; then
                read LIP
            fi
            read RID_TYPE
            read RID
            read RIP
            read POLICY
            read KEY
            read AUTOSTART
            echo "ikegui 1 1 2 0 $NAME $LID_TYPE \"$LID\" $LIP $RID_TYPE \"$RID\" \
$RIP $POLICY $KEY $AUTOSTART > /dev/null 2>&1"
            PrintDot
        else
            return 0
        fi
    done
}

function P2TunRestore {
    while :
    do
        read TUNID
        read NAME
        if [[ $? = 0 ]]; then
            read P1TUN
            read LTYPE
            read LID
            read LMASK
            read LPROT
            read LPORT
            read RTYPE
            read RID
            read RMASK
            read RPROT
            read RPORT
            read POLICY
            read AUTOSTART
            echo "ikegui 1 2 2 0 $NAME $P1TUN $LTYPE $LID $LMASK $LPROT $LPORT $RTYPE
              \$RID $RMASK $RPROT $RPORT $POLICY $AUTOSTART > /dev/null 2>&1"
            PrintDot
        else
            return 0
        fi
    done
}

function allRestoreWithIkedb {

    ERRORS=/tmp/ikedb_msgs.bos.net.ipsec.keymgt
    echo > $ERRORS
    $IKEDB -p $XMLFILE 2>> $ERRORS
    if [ -f $PSKXMLFILE ]
    then
        $IKEDB -p $PSKXMLFILE 2>> $ERRORS
    fi

}

P1PROPFILE=/tmp/p1proposal.bos.net.ipsec.keymgt
P2PROPFILE=/tmp/p2proposal.bos.net.ipsec.keymgt
P1POLFILE=/tmp/p1policy.bos.net.ipsec.keymgt
P2POLFILE=/tmp/p2policy.bos.net.ipsec.keymgt
P1TUNFILE=/tmp/p1tunnel.bos.net.ipsec.keymgt
P2TUNFILE=/tmp/p2tunnel.bos.net.ipsec.keymgt
XMLFILE=/tmp/full_ike_database.bos.net.ipsec.keymgt
PSKXMLFILE=/tmp/psk_ike_database.bos.net.ipsec.keymgt
CMD_FILE=/tmp/commands
IKEDB=$(which ikedb) || IKEDB=/usr/sbin/ikedb

echo "building ISAKMP database \n"
$IKEDB -x || exit $?

if [ -f $XMLFILE ]; then
    echo "\nRestoring database entries\c"
    allRestoreWithIkedb
    echo "\ndone\n"

elif [ -f /tmp/*.bos.net.ipsec.keymgt ]; then
    echo "\nRestoring database entries\c"

    LPPLEVEL=`cat /tmp/lpplevel`

    echo > $CMD_FILE
    touch $P1PROPFILE; P1PropRestore < $P1PROPFILE >> $CMD_FILE
    touch $P2PROPFILE; P2PropRestore < $P2PROPFILE >> $CMD_FILE
    touch $P1POLFILE; P1PolRestore < $P1POLFILE >> $CMD_FILE
    touch $P2POLFILE; P2PolRestore < $P2POLFILE >> $CMD_FILE
    touch $P1TUNFILE; P1TunRestore < $P1TUNFILE >> $CMD_FILE
    touch $P2TUNFILE; P2TunRestore < $P2TUNFILE >> $CMD_FILE

    mv $P1PROPFILE ${P1PROPFILE}.loaded
    mv $P2PROPFILE ${P2PROPFILE}.loaded
    mv $P1POLFILE ${P1POLFILE}.loaded
    mv $P2POLFILE ${P2POLFILE}.loaded
    mv $P1TUNFILE ${P1TUNFILE}.loaded
    mv $P2TUNFILE ${P2TUNFILE}.loaded

    ksh $CMD_FILE

    echo "done\n"
fi
```

Network Information Services and NIS+ security

This section provides an overview of how NIS+ protects its namespace.

This section includes the following sections:
- "Operating system security mechanisms"
- "NIS+ Security mechanisms" on page 200
- "NIS+ Authentication and credentials" on page 202
- "NIS+ authorization and access" on page 204
- "NIS+ Security and administrative rights" on page 207
- "NIS+ security reference" on page 207

Operating system security mechanisms

Operating system security is provided by gates that users must pass through before entering the operating system environment, and permission matrixes that determine what they are able to do once inside. In some contexts, *secure RPC* passwords have been referred to as *network passwords*.

The overall system is composed of four gates and two permission matrixes:

Dialup gate
To access a given operating system environment from the outside through a modem and phone line, you must provide a valid login ID and dial-up password.

Login gate
To enter a given operating system environment you must provide a valid login ID and user password.

Root gate
To gain access to root privileges, you must provide a valid root user password.

Secure RPC gate
In an NIS+ environment running at security level 2 (the default), when you try to use NIS+ services and gain access to NIS+ objects (servers, directories, tables, table entries, and so on) your identity is confirmed by NIS+, using the secure RPC process.

Entering the secure RPC gate requires presentation of a secure RPC password. Your secure RPC password and your login password normally are identical. When that is the case, you are passed through the gate automatically without having to re-enter your password. (In some contexts, *secure RPC* passwords have been referred to as *network passwords*. See the Secure RPC Password versus Login Password section in the *AIX 5L Version 5.3 Network Information Services (NIS and NIS+) Guide* for information about handling two passwords that are not identical.)

A set of *credentials* is used to automatically pass your requests through the secure RPC gate. The process of generating, presenting, and validating your credentials is called *authentication* because it confirms who you are and that you have a valid secure RPC password. This authentication process is automatically performed every time you request NIS+ service.

In an NIS+ environment running in NIS-compatibility mode, the protection provided by the secure RPC gate is significantly weakened because everyone has read rights for all NIS+ objects and modify rights for those entries that apply to them regardless of whether or not they have a valid credential (that is, regardless of whether or not the authentication process has confirmed their identity and validated their secure RPC password). Because this situation allows *anyone* to have read rights for all NIS+ objects and modify rights for those entries that apply to them, an NIS+ network running in compatibility mode is less secure than one running in normal mode. (In secure RPC terminology, any user without a valid credential is considered a member of the nobody class. See "Authorization classes" on page 204 for a description of the four classes.)

For details on how to administer NIS+ authentication and credentials, see the Administering NIS+ Credentials section in the *AIX 5L Version 5.3 Network Information Services (NIS and NIS+) Guide*.

File and directory matrix

Once you have gained access to an operating system environment, your ability to read, execute, modify, create, and destroy files and directories is governed by the applicable permissions.

NIS+ objects matrix

Once you have been properly authenticated to NIS+, your ability to read, modify, create, and destroy NIS+ objects is governed by the applicable permissions. This process is called *NIS+ authorization*.

For details on NIS+ permissions and authorization, see the Administering NIS+ Access Rights section in the *AIX 5L Version 5.3 Network Information Services (NIS and NIS+) Guide*.

NIS+ Security mechanisms

NIS+ security is an integral part of the NIS+ namespace. You cannot set up security independently from the namespace. For this reason, instructions for setting up security are woven through the steps used to set up the other components of the namespace.

Once an NIS+ security environment has been set up, you can add and remove users, change permissions, reassign group members, and perform all other routine administrative tasks needed to manage an evolving network.

The security features of NIS+ protect the information in the namespace, as well as the structure of the namespace itself, from unauthorized access. Without these security features, any NIS+ client could obtain, change, or even damage information stored in the namespace.

NIS+ security serves two purposes:

Authentication

Authentication is used to identify NIS+ principals. Every time a principal (either user or machine) tries to access an NIS+ object, the user's identity and secure RPC password is confirmed and validated. (You should not have to enter a password as part of the authentication process. However, if for some reason your secure RPC password is different from your login password, you must perform a **keylogin** the first time you try accessing NIS+ objects or services. To perform a **keylogin**, you must provide a valid secure RPC password. See the Secure RPC Password versus Login Password section in the *AIX 5L Version 5.3 Network Information Services (NIS and NIS+) Guide*.)

Authorization

Authorization is used to specify access rights. Every time NIS+ principals try to access NIS+ objects, they are placed in one of four authorization classes (owner, group, world, nobody). The NIS+ security system allows NIS+ administrators to specify different read, modify, create, or destroy rights to NIS+ objects for each class. For example, a given class could be permitted to modify a particular column in the passwd table but not read that column, or a different class could be allowed to read some entries of a particular table but not others.

For example, a given NIS+ table might allow one class to both read and modify the information in the table, but a different class is only allowed to read the information, and a third class is not even allowed to do that. This is similar in concept to the operating system's file and directory permissions system. (See "Authorization classes" on page 204 for more information on classes.)

Authentication and authorization prevents someone with root privileges on machine A from using the **su** command to assume the identity of a second user who is either not logged in at all or logged in on machine B, and then accessing NIS+ objects with the second user's NIS+ access privileges.

Note, however, that NIS+ cannot prevent someone who knows another user's login password from assuming that other user's identity and NIS+ access privileges. Nor can NIS+ prevent a user with root privileges from assuming the identity of another user who is logged in from the *same* machine.

The following figure details the NIS+ security process.

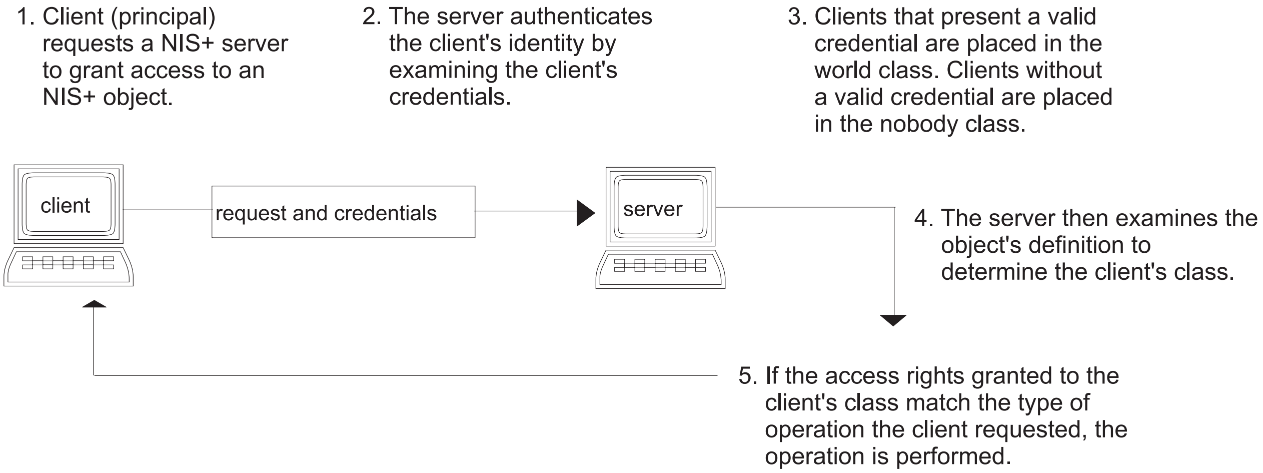

1. Client (principal) requests a NIS+ server to grant access to an NIS+ object.
2. The server authenticates the client's identity by examining the client's credentials.
3. Clients that present a valid credential are placed in the world class. Clients without a valid credential are placed in the nobody class.
client
request and credentials
server
4. The server then examines the object's definition to determine the client's class.
5. If the access rights granted to the client's class match the type of operation the client requested, the operation is performed.

Figure 15. Summary of NIS+ Security Process

1. The client or principal requests an NIS+ server to grant access to an NIS+ object.
2. The server authenticates the client's identity by examining the client's credentials.
3. The clients with valid credentials are placed in the world class.
4. The clients without valid credentials are placed in the nobody class.
5. The server examines the object's definition to determine the client's class.
6. If the access rights granted to the client's class match the type of operation requested, the operation is performed.

NIS+ Principals

NIS+ principals are the entities (clients) that submit requests for NIS+ services. An NIS+ principal might be someone who is logged in to a client machine as a regular user, someone who is logged in as root user, or any process that runs with root user permission on an NIS+ client machine. Thus, an NIS+ principal can be a client user or a client workstation.

An NIS+ principal can also be the entity that supplies an NIS+ service from an NIS+ server. Because all NIS+ servers are also NIS+ clients, much of the information in this section also applies to servers.

NIS+ Security levels

NIS+ servers operate at one of two security levels. These levels determine the types of credential principals must submit for their requests to be authenticated.

NIS+ is designed to run at the most secure level, which is security level 2. Level 0 is provided only for testing, setup, and debugging purposes. These security levels are summarized in the following table.

Note: Use Web-based System Manager, SMIT, or the **passwd** command to change your own password regardless of security level or credential status.

NIS+ security levels

Severity level	Description
0	Security level 0 is designed for testing and setting up the initial NIS+ namespace. An NIS+ server running at security level 0 grants any NIS+ principal full access rights to all NIS+ objects in the domain. Level 0 is for setup purposes only and should only be used by administrators for that purpose. Level 0 should *not* be used on networks in normal operation by regular users.
1	Security level 1 uses AUTH_SYS security. This level is not supported by NIS+ and should *not* be used.
2	Security level 2 is the default. The highest level of security currently provided by NIS+, it authenticates only requests that use data encryption standard (DES) credentials. Requests with no credentials are assigned to the nobody class and have whatever access rights have been granted to that class. Requests that use invalid DES credentials are retried. After repeated failure to obtain a valid DES credential, requests with invalid credentials fail with an authentication error. (A credential might not be valid for a variety of reasons, such as the principal making the request is not logged in through **keylogin** on that machine, the clocks are out of sync, there is a key mismatch, and so on.)

NIS+ Authentication and credentials

NIS+ credentials authenticate the identity of each principal requesting an NIS+ service or access to an NIS+ object.

The NIS+ credential or authorization process is an implementation of the Secure RPC system.

The credential or authentication system prevents someone from assuming another's identity. That is, it prevents someone with root privileges on one machine from using the **su** command to assume the identity of a second user who is either not logged in at all or logged in on another machine and then accessing NIS+ objects with the second user's NIS+ access privileges.

Note: NIS+ cannot prevent someone who knows another user's login password from assuming that other user's identity and the other user's NIS+ access privileges. Nor can NIS+ prevent a user with root privileges from assuming the identity of another user who is currently logged in on the *same* machine.

After a server authenticates a principal, it then checks the NIS+ object that the principal wants to access to verify what operations that principal is authorized to perform. (For further information about authorization, see "NIS+ authorization and access" on page 204.)

User and machine credentials

This section explains user and machine credentials.

For the basic types of principal, *users* and *machines,* the following different types of credentials exist:

User credentials
When someone is logged in to an NIS+ client as a regular user, requests for NIS+ services include that person's user credentials.

Machine credentials
When a user is logged in to an NIS+ client as root user, request for services use the client workstation's credentials.

DES versus local credentials

NIS+ principals can have DES or local credentials.

DES credentials:

Data Encryption Standard (DES) credentials provide secure authentication.

When this guide refers to NIS+ checking a credential to authenticate an NIS+ principal, it is the DES credential that NIS+ is validating.

Note: Using DES credentials is only one method of achieving authentication. Do not equate DES credentials with NIS+ credentials.

Each time a principal requests an NIS+ service or access to an NIS+ object, the software uses the credential information stored for that principal to generate a credential for that principal. DES credentials are generated from information created for each principal by an NIS+ administrator, as explained in the Administering NIS+ Credentials section in the *AIX 5L Version 5.3 Network Information Services (NIS and NIS+) Guide*.

- When the validity of a principal's DES credential is confirmed by NIS+, that principal is *authenticated*.
- A principal must be authenticated before being placed in the owner, group, or world authorization classes. In other words, you must have a valid DES credential in order to be placed in one of those classes. (Principals without a valid DES credential are automatically placed in the nobody class.)
- DES credential information is always stored in the cred table of the principal's home domain, regardless of whether that principal is a client user or a client workstation.

Local credentials:

Local credentials are a map between a user's user ID number and their NIS+ principal name, which includes their home domain name.

When users log in, the system looks up their local credential, which identifies their home domain where their DES credential is stored. The system uses that information to get the user's DES credential information.

When users log in to a remote domain, those requests use their local credential, which points back to their home domain. NIS+ then queries the user's home domain for that user's DES credential information. This allows a user to be authenticated in a remote domain even though the user's DES credential information is not stored in that domain. The following figure illustrates this concept.

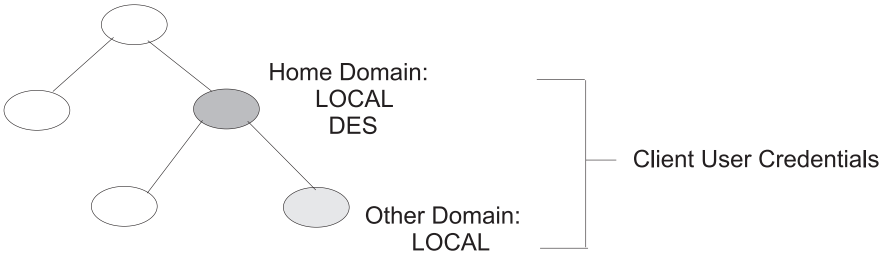


Figure 16. Credential and Domains

This illustration shows a domain hierarchy. The user's home domain has local and DES credentials. The subdomain only has local credentials. Home and the subdomain are labeled Client User Credentials.

Local credential information can be stored in any domain. To log in to a remote domain and be authenticated, a client user *must* have a local credential in the cred table of the remote domain. If a user does not have a local credential in a remote domain the user is trying to access, NIS+ cannot locate the user's home domain to obtain the user's DES credential. In such a case, the user would not be authenticated and would be placed in the nobody class.

User types and credential types:

A user can have both types of credential, but a machine can *only* have a DES credential.

Root cannot have NIS+ access, as root, to other machines because the root UID of every machine is always zero. If root (UID=0) of machine A tried to access machine B as root user, that conflicts with machine B's already existing root (UID=0). Thus, a local credential is not appropriate for a client *workstation*; it is allowed only for a client *user*.

NIS+ authorization and access

The basic purpose of NIS+ authorization is to specify the access rights that each NIS+ principal has for each NIS+ object and service.

After the principal making an NIS+ request is authenticated, NIS+ places that principal in an authorization class. The access rights (permissions) that specify which operations a principal may do with a given NIS+ object are assigned on a class basis. In other words, one authorization class may have certain access rights while a different class has different rights.

Authorization classes
The following authorization classes exist: owner, group, world, and nobody. (See "Authorization classes" for details.)

Access rights
The following types of access rights (permissions) exist: create, destroy, modify, and read. (See "NIS+ access rights" on page 207 for details.)

Authorization classes

NIS+ objects do not grant access rights directly to NIS+ principals.

NIS+ objects grant access rights to the following *classes* of principal:

Owner
The principal who happens to be the object's owner gets the rights granted to the owner class.

Group Each NIS+ object has one group associated with it. The members of an object's group are specified by the NIS+ administrator. The principals who belong to the object's group class get the rights granted to the group class. (In this context, *groups* refers to NIS+ groups, and not operating system or net groups. For a description of NIS+ groups, see "Group class" on page 205.

World The world class encompasses all NIS+ principals that a server has been able to authenticate. (That is, everyone who has been authenticated but who is not in either the owner or group classes.)

Nobody
All principals belong to the nobody class, including those who are not authenticated.

The following figure illustrates the class relationship:

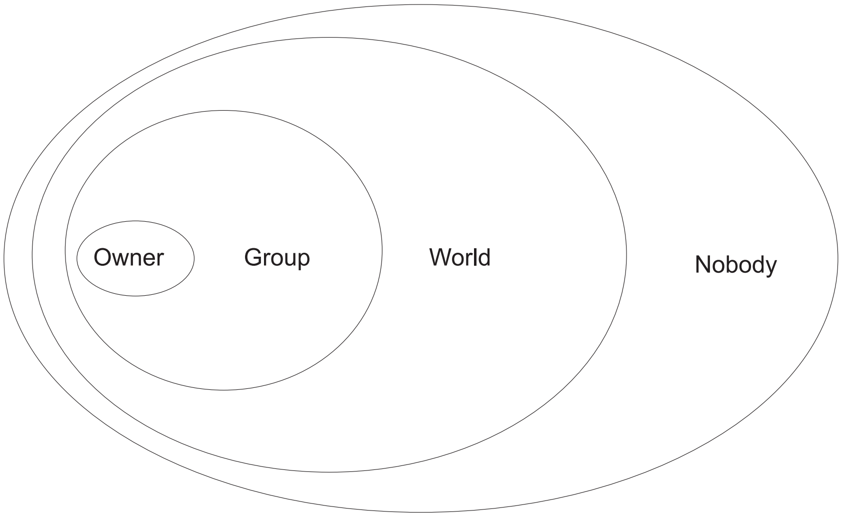


Figure 17. Authorization Classes

This illustration shows a series of ovals within ovals that represents the relationship between authorization classes. The smallest oval is Owner, encompassed by a larger oval labeled Group, encompassed by an oval labeled World, encompassed by an oval labeled Nobody.

For any NIS+ request, the system determines which class the requesting principal belongs to and the principal can then use whatever access rights belong to that class.

An object can grant any combination of access rights to each of these classes. Normally, however, a higher class is assigned the same rights as all the lower classes, as well as possible additional rights.

For instance, an object could grant read access to the nobody and world classes, both read and modify access to the group class, and read, modify, create, and destroy access to the owner class.

The following section describes the authorization classes in detail:

Owner class:

The owner is a *single* NIS+ principal.

A principal making a request for access to an NIS+ object must be authenticated (present a valid DES credential) before being granted owner-access rights.

By default, an object's owner is the principal that created the object. However, an object's owner can cede ownership to another principal by two different methods:

- The principal specifies a different owner at the time the object is created (see the Specifying Access Rights in Commands section in the *AIX 5L Version 5.3 Network Information Services (NIS and NIS+) Guide*).
- The principal changes the ownership of the object after it is created (see the Changing Ownership of Objects and Entries section in the *AIX 5L Version 5.3 Network Information Services (NIS and NIS+) Guide*).

After a principal cedes ownership, that principal cedes all owner's access rights to the object and keeps only the rights the object assigns to either the group, the world, or nobody.

Group class:

The object's group is a *single* NIS+ group. (In this context, *group* refers to NIS+ groups, and not operating system or net groups.)

A principal making a request for access to an NIS+ object must be authenticated (present a valid DES credential) and belong to the group before being granted group-access rights.

An NIS+ group is a collection of NIS+ principals, grouped together as a convenience for providing access to the namespace. The access rights granted to an NIS+ group apply to all the principals that are members of that group. (An object's owner, however, does not need to belong to the object's group.)

When an object is created, the creator can opt for a default group. A nondefault group can be specified either when the object is created or at any time later.

Information about NIS+ groups is stored inNIS+ group **objects**, under the groups_dir subdirectory of every NIS+ domain. (Note that information about NIS+ groups is not stored in the NIS+ group table. That table stores information about operating system groups.) Instructions for administering NIS+ groups are provided in the Administering NIS+ Groups section in the *AIX 5L Version 5.3 Network Information Services (NIS and NIS+) Guide*.

World class:

The world class contains all NIS+ principals that are authenticated by NIS+; that is, all members in the owner and group class, as well as all other principals who present a valid DES credential.

Access rights granted to the world class apply to all authenticated principals.

Nobody class:

The nobody class contains all principals, even those without a valid DES credential.

Authorization classes and the NIS+ object hierarchy:

NIS+ security applies authorization classes independently to a hierarchy of objects.

Directory objects are the top level of the default hierarchy, then group or table objects, then columns, then entries. The following definitions provide more information about each level:

Directory level
Each NIS+ domain contains two NIS+ directory objects: groups_dir and org_dir. Each groups_dir directory object contains various groups. Each org_dir directory object contains various tables.

Group or table level
Groups contain individual entries and possibly other groups. Tables contain both columns and individual entries.

Column level
Each table has one or more columns.

Entry (row) level
Each group or table has one or more entries.

The four authorization classes apply at each level. Thus, a directory object has an owner and a group. Each table within a directory object has its own owner and group, which can be different from the owner and group of the directory object. Within a table, a column or an entry can have its own owner or group, which can be different from the owner and group of the table as a whole or of the directory object as a whole.

NIS+ access rights

NIS+ objects specify access rights for NIS+ principals in the same way that operating system files specify permissions for operating system users.

Access rights specify the types of operations that NIS+ principals are allowed to perform on NIS+ objects. (You can examine these by using the **niscat -o** command.)

NIS+ operations vary among different types of objects, but all operations fall into one of the following access-rights categories: read, modify, create, and destroy.

Read A principal with read rights to an object can view the contents of that object.

Modify
A principal with modify rights to an object can change the contents of that object.

Destroy
A principal with destroy rights to an object can destroy or delete the object.

Create
A principal with create rights to a higher-level object can create new objects within that level. If you have create rights to a NIS+ directory object, you can create new tables within that directory. If you have create rights to a NIS+ table, you can create new columns and entries within that table.

Every communication from an NIS+ client to an NIS+ server is a request to perform one of these operations on a specific NIS+ object. For instance, when an NIS+ principal requests the IP address of another workstation, it is effectively requesting read access to the hosts table object, which stores that type of information. When a principal asks the server to add a directory to the NIS+ namespace, it is actually requesting modify access to the directory's parent object.

These rights logically evolve down from directory to table to table column and entry levels. For example, to create a new table, you must have create rights for the NIS+ directory object where the table will be stored. When you create that table, you become its default owner. As owner, you can assign yourself create rights to the table, which allows you to create new entries in the table. If you create new entries in a table, you become the default owner of those entries. As table owner, you can also grant table-level create rights to others. For example, you can give your table's group class table-level create rights. In that case, any member of the table's group can create new entries in the table. The individual member of the group who creates a new table entry becomes the default owner of that entry.

NIS+ Security and administrative rights

NIS+ does not enforce any requirement that there be a single NIS+ administrator. Whoever has administrative rights over an object (that is, the authority to create, destroy, and for some objects, modify rights) is considered to be an NIS+ administrator for that object.

Whoever creates an NIS+ object sets the initial access rights to that object. If the creator restricts administrative rights to the object's owner (initially the creator), only the owner has administrative power over that object. On the other hand, if the creator grants administrative rights to the object's group, then everyone in that group has administrative power over that object.

Theoretically, you could grant administrative rights to the world class, or even the nobody class. The software allows you to do that. But granting administrative rights beyond the group class effectively nullifies NIS+ security. Thus, if you grant administrative rights to either the world or the nobody class, you are, in effect, defeating the purpose of NIS+ security.

NIS+ security reference

This section provides a list of commands to administer passwords, credentials, and keys.

chkey Changes a principal's secure RPC key pair. Unless you want to re-encrypt your current private key with a new password, use the **passwd** command instead. The **chkey** command does not affect the principal's entry either in the passwd table or **/etc/passwd** file.

keylogin
Decrypts and stores a principal's secret key with the keyserv.

keylogout
Deletes stored secret key from keyserv.

keyserv
Enables the server for storing private encryption keys.

newkey
Creates a new key pair in public-key database.

nisaddcred
Creates credentials for NIS+ principals.

nisupdkeys
Updates public keys in directory objects.

passwd
Changes and administers principal's password.

Network File System security

The Network File System (NFS) is a widely available technology that allows data to be shared between various hosts on a network.

Beginning with the release of AIX 5.3.0, NFS also supports the use of Kerberos 5 authentication in addition to DES. Kerberos 5 security is provided under a protocol mechanism called RPCSEC_GSS. For additional information on Kerberos authentication with NFS, see Setting up a network for RPCSEC-GSS in *AIX 5L Version 5.3 System Management Guide: Communications and Networks*.

For general information about NFS support on AIX 5L Version 5.1, refer to Chapter 10. Network File System in the System Management Guide: Communications and Networks.

For general information about NFS support on AIX 5L Version 5.2, refer to the Network File System chapter in the System Management Guide: Communications and Networks.

In addition to the standard UNIX authentication system, NFS provides a means to authenticate users and machines in networks on a message-by-message basis. This additional authentication system uses Data Encryption Standard (DES) encryption and public key cryptography.

Beginning with the release of AIX 5L Version 5.2, NFS also supports the use of Kerberos 5 authentication in addition to DES. Kerberos 5 security is provided under a protocol mechanism called RPCSEC_GSS. For a description of how to administer and use Kerberos authentication with NFS, see the *NFS Administration Guide*.

General guidelines for securing Network File System

The following information consists of guidelines to help you secure NFS:

- Ensure that the latest software patches are installed. Patches that address security issues should be considered especially important. All software in a given infrastructure should be maintained. For example, installing patches in an operating system but failing to install patches on a Web server may provide an attacker with a way to attach your environment that could have been avoided if the Web server been updated as well. To subscribe to pSeries Security Alerts for information about the latest available security information, visit the following Web address: https://techsupport.services.ibm.com/server/pseries.subscriptionSvcs.

- Configure the NFS server to export file systems with the least amount of privileges necessary. If users only need to read from a file system, they should not be able to write to the file system. This can mitigate an attempt to overwrite important data, modify configuration files, or write malicious executable code to an exported file system. Specify privileges using SMIT or by directly editing the **/etc/exports** file.
- Configure the NFS server to export file systems explicitly for the users who should have access to it. Most implementations of NFS will allow you to specify which NFS clients should have access to a given file system. This will mitigate attempts by unauthorized users to access file systems. In particular, do not configure a NFS server to export a file system to itself.
- Exported file systems should be in their own partitions. An attacker could cause system degradation by writing to an exported file system until it is full. This may make the file system unavailable to other applications or users that needed it.
- Do not allow NFS clients to access the file system with root user credentials or unknown user credentials. Most implementations of NFS can be configured to map requests from a privileged or unknown user to an unprivileged user. This will avert scenarios where an attacker tries to access files and perform file operations as a privileged user.
- Do not allow NFS clients to execute suid and sgid programs on exported file systems. This will prevent NFS clients from executing malicious code with privileges. If the attacker is able to make the executable owned by a privileged owner or group, significant harm can be done to the NFS server. This can be done by specifying the **mknfsmnt -y** command option.
- Use Secure NFS. Secure NFS uses DES encryption to authenticate hosts involved in RPC transactions. RPC is a protocol used by NFS to communicate requests between hosts. Secure NFS will mitigates attempts by an attacker to spoof RPC requests by encrypting the time stamp in the RPC requests. A receiver successfully decrypting the time stamp and confirm that it is correct serves as confirmation that the RPC request came from a trusted host.
- If NFS is not needed, turn it off. This will reduce the number of possible attack vectors available to an intruder.

For more information about implementing the items discussed, refer to the following information:

- **AIX 5L Version 5.1 information**
  - NFS Installation and Configuration: http://publibn.boulder.ibm.com/doc_link/en_US/a_doc_lib/aixbman/commadmn/nfs_install.htm
  - exports File for NFS: http://publibn.boulder.ibm.com/doc_link/en_US/a_doc_lib/files/aixfiles/exports.htm
  - Secure NFS: http://publibn.boulder.ibm.com/doc_link/en_US/a_doc_lib/aixbman/commadmn/nfs_secure.htm
  - mknfsmnt Command: http://publibn.boulder.ibm.com/doc_link/en_US/a_doc_lib/cmds/aixcmds3/mknfsmnt.htm
- **AIX 5L Version 5.2 information**
  - NFS Installation and Configuration: http://publib16.boulder.ibm.com/pseries/en_US/aixbman/commadmn/nfs_install.htm
  - exports File for NFS: http://publib16.boulder.ibm.com/pseries/en_US/files/aixfiles/exports.htm
  - Network File System (NFS) Security: http://publib16.boulder.ibm.com/pseries/en_US/aixbman/security/secure_nfs.htm
  - mknfsmnt Command: http://publib16.boulder.ibm.com/pseries/en_US/cmds/aixcmds3/mknfsmnt.htm

Network File System authentication

NFS uses the DES algorithm for different purposes. NFS uses DES to encrypt a time stamp in the remote procedure call (RPC) messages sent between NFS servers and clients. This encrypted time stamp authenticates machines just as the token authenticates the sender.

Because NFS can authenticate every RPC message exchanged between NFS clients and servers, this provides an additional, optional level of security for each file system. By default, file systems are exported with the standard UNIX authentication. To take advantage of this additional level of security, you can specify the secure option when you export a file system.

Public key cryptography for secure Network File System

Both the public key and the secret key of the user are stored and indexed by the net name in the **publickey.byname** map.

The secret key is DES-encrypted with the user login password. The **keylogin** command uses the encrypted secret key, decrypts it with the login password, then gives it to a secure local key server to save for use in future RPC transactions. Users are not aware of their public and secret keys because the **yppasswd** command, in addition to changing the login password, generates the public and secret keys automatically.

The keyserv daemon is an RPC service that runs on each NIS and NIS+ machine. For information on how NIS+ uses **keyserv**, see *AIX 5L Version 5.3 Network Information Services (NIS and NIS+) Guide*. Within NIS, **keyserv** executes the following public key subroutines:

- **key_setsecret** subroutine
- **key_encryptsession** subroutine
- **key_decryptsession** subroutine

The **key_setsecret** subroutine tells the key server to store the secret key of the user (SK_A) for future use; it is normally called by the **keylogin** command. The client program calls the **key_encryptsession** subroutine to generate the encrypted conversation key, which is passed in the first RPC transaction to a server. The key server looks up the server public key and combines it with the secret key of the client (set up by a previous **key_setsecret** subroutine) to generate the common key. The server asks the key server to decrypt the conversation key by calling the **key_decryptsession** subroutine.

Implicit in these subroutine calls is the name of the caller, which must be authenticated in some manner. The key server cannot use DES authentication to do this, because it would create a deadlock. The key server solves this problem by storing the secret keys by the user ID (UID) and only granting requests to local root processes. The client process then executes a root-user-owned **setuid** subroutine that makes the request on the part of the client, telling the key server the real UID of the client.

Network File System authentication requirements

Secure NFS authentication is based on the ability of a sender to encrypt the current time, which the receiver can then decrypt and check against its own clock.

This process has the following requirements:

- The two agents must agree on the current time.
- The sender and receiver must be using the same DES encryption key.

Agreeing on the current time:

If the network uses time synchronization, the timed daemon keeps the client and server clocks synchronized. If not, the client computes the proper time stamps based on the server clock.

To do this, the client determines the server time before starting the RPC session, and then computes the time difference between its own clock and that of the server. The client then adjusts its time stamp accordingly. If, during the course of an RPC session, the client and server clocks become unsynchronized to the point where the server begins rejecting the client requests, the client will redetermine the server time.

Using the same DES key:

The client and server compute the same DES encryption key by using public key cryptography.

For any client A and server B, a key called the *common key* can only be deduced by A and B. This key is . The client derives the common key by computing the following formula:

$K_{AB} = PK_B^{SK}A$

where *K* is the common Key, *PK* is the Public Key, and *SK* is the Secret Key, and each of these keys is a 128-bit number. The server derives the same common key by computing the following formula:

$K_{AB} = PK_A^{SK}B$

Only the server and client can calculate this common key since doing so requires knowing one secret key or the other. Because the common key has 128 bits, and DES uses a 56-bit key, the client and server extract 56 bits from the common key to form the DES key.

Network File System authentication process

When a client wants to talk to a server, it randomly generates a key used for encrypting the time stamps. This key is known as the *conversation key* (*CK*).

The client encrypts the conversation key using the DES common key (described in Authentication Requirements) and sends it to the server in the first RPC transaction. This process is illustrated in the following figure.

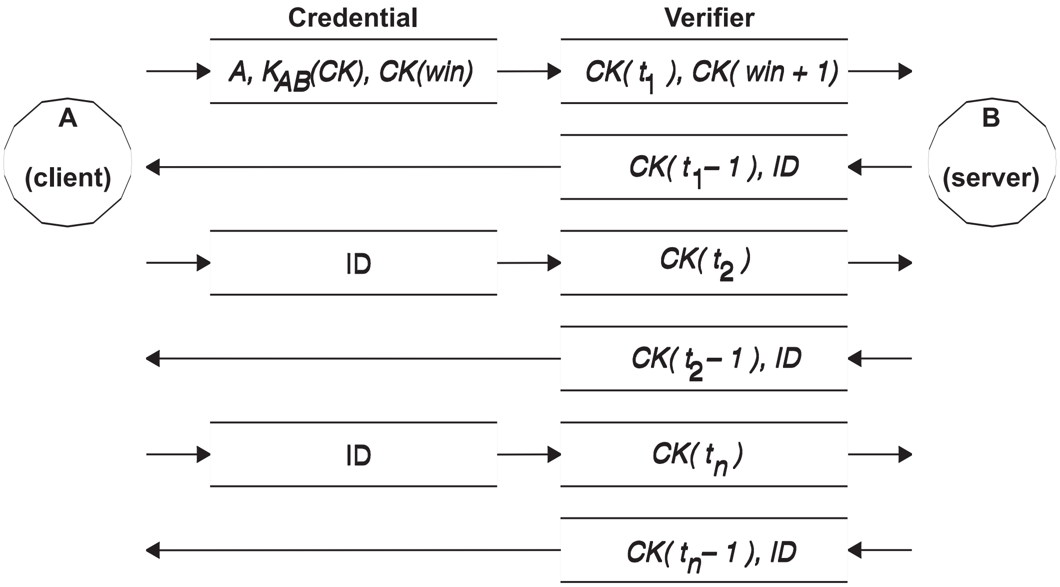


Figure 18. Authentication Process. This figure illustrates the authentication process.

This figure shows client A connecting to server B. The term *K*(*CK*) means *CK* is encrypted with the DES common key *K.* In its first request, the client RPC credential contains the client name (*A*), the conversation key (*CK*), and the variable called *win* (window) encrypted with *CK*. (The default window size is 30 minutes.) The client verifier in the first request contains the encrypted time stamp and an encrypted verifier of the specified window, *win* + 1. The window verifier makes guessing the right credential much more difficult, and increases security.

After authenticating the client, the server stores the following items in a credential table:

- Client name, *A*

- Conversation key, *CK*
- Window
- Time stamp

The server only accepts time stamps that are chronologically greater than the last one seen, so any replayed transactions are guaranteed to be rejected. The server returns to the client in the verifier an index ID into the credential table, plus the client time stamp minus 1, encrypted by *CK*. The client knows that only the server could have sent such a verifier, because only the server knows what time stamp the client sent. The reason for subtracting 1 from the time stamp is to ensure that it is not valid and cannot be reused as a client verifier. After the first RPC transaction, the client sends just its ID and an encrypted time stamp to the server, and the server sends back the client time stamp minus 1, encrypted by *CK*.

Naming network entities for DES authentication

DES authentication does its naming by using net names. For information on how NIS+ handles DES authentication, see the *AIX 5L Version 5.3 Network Information Services (NIS and NIS+) Guide*.

A *net name* is a string of printable characters to authenticate. The public and secret keys are stored on a per-net-name rather than a per-user-name basis. The **netid.byname** NIS map maps the net name into a local UID and group-access list.

User names are unique within each domain. Net names are assigned by concatenating the operating system and user ID with the NIS and Internet domain names. A good convention for naming domains is to append the Internet domain name (com, edu, gov, mil) to the local domain name.

Network names are assigned to machines as well as to users. A net name of a machine is formed much like that of a user. For example, a machine named `hal` in the `eng.xyz.com` domain has the net name `unix.hal@eng.xyz.com`. Correct authentication of machines is important for diskless machines that need full access to their home directories over the network.

To authenticate users from any remote domain, make entries for them in two NIS databases. One is an entry for their public and secret keys; the other is for their local UID and group-access list mapping. Users in the remote domain can then access all of the local network services, such as the NFS and remote logins.

The /etc/publickey file

The **/etc/publickey** file contains names and public keys, which NIS and NIS+ use to create the publickey map.

The publickey map is used for secure networking. Each entry in the file consists of a network user name (which refers to either a user or a host name), followed by the user public key (in hexadecimal notation), a colon, and the user-encrypted secret key (also in hexadecimal notation). By default, the only user in the **/etc/publickey** file is the user nobody.

Do not use a text editor to alter the **/etc/publickey** file because the file contains encryption keys. To alter the **/etc/publickey** file, use either the **chkey** or **newkey** commands.

Public key systems booting considerations

When restarting a machine after a power failure, all of the stored secret keys are lost, and no process can access secure network services, such as mounting an NFS. Root processes could continue if there were someone to enter the password that decrypts the secret key of the root user. The solution is to store the root-user decrypted secret key in a file that the key server can read.

Not all **setuid** subroutine calls operate correctly. For example, if a **setuid** subroutine is called by owner `A`, and owner `A` has not logged into the machine since it started, the subroutine cannot access any secure network services as `A`. However, most **setuid** subroutine calls are owned by the root user, and the root user secret key is always stored at startup time.

Secure Network File System performance considerations

Secure NFS affects system performance in the following ways:

- Both the client and server must compute the common key. The time it takes to compute the common key is about one second. As a result, it takes about two seconds to establish the initial RPC connection, because both client and server have to perform this operation. After the initial RPC connection, the key server caches the results of previous computations, and so it does not have to recompute the common key every time.
- Each RPC transaction requires the following DES encryption operations:
  1. The client encrypts the request time stamp.
  2. The server decrypts it.
  3. The server encrypts the reply time stamp.
  4. The client decrypts it.

Because system performance can be reduced by secure NFS, weigh the benefits of increased security against system-performance requirements.

Secure Network File System checklist

Use the following checklist to help ensure that secure NFS operates correctly:

- When mounting a file system with the **-secure** option on a client, the server name must match the server host name in the **/etc/hosts** file. If a name server is being used for host-name resolution, make sure the host information returned by the name server matches the entry in the **/etc/hosts** file. Authentication errors result if these names do not match because the net names for machines are based on the primary entries in the **/etc/hosts** file and keys in the **publickey** map are accessed by net name.
- Do not mix secure and nonsecure exports and mounts. Otherwise, file access might be determined incorrectly. For example, if a client machine mounts a secure file system without the **-secure** option or mounts an nonsecure system with the **-secure** option, users have access as `nobody`, rather than as themselves. This condition also occurs if a user unknown to NIS or NIS+ attempts to create or modify files on a secure file system.
- Because NIS must propagate a new map after each use of the **chkey** and **newkey** commands, use these commands only when the network is lightly loaded.
- Do not delete the **/etc/keystore** file or the **/etc/.rootkey** file. If you reinstall, move, or upgrade a machine, save the **/etc/keystore** and **/etc/.rootkey** files.
- Instruct users to use the **yppasswd** command rather than the **passwd** command to change passwords. Doing so keeps passwords and private keys synchronized.
- Because the **login** command does not retrieve keys out of the publickey map for the **keyserv** daemon, the user must execute the **keylogin** command. You may want to place the **keylogin** command in each user **profile** file to execute the command automatically during login. The **keylogin** command requires users to enter their password again.
- When you generate keys for the root user at each host with either the **newkey -h** or **chkey** command, you must run the **keylogin** command to pass the new keys to the **keyserv** daemon. The keys are stored in the **/etc/.rootkey** file, which is read by the **keyserv** daemon each time the daemon is started.
- Periodically verify that the **yppasswdd** and **ypupdated** daemons are running on the NIS master server. These daemons are necessary for maintaining the publickey map.
- Periodically verify that the **keyserv** daemon is running on all machines using secure NFS.

Configuring secure Network File System

To configure secure NFS on NIS master and slave servers, use the Web-based System Manager Network application or use the following procedure.

For information about using NFS with NIS+, see *AIX 5L Version 5.3 Network Information Services (NIS and NIS+) Guide*.

1. On the NIS master server, create an entry for each user in the NIS **/etc/publickey** file by using the **newkey** command as follows:
   - For a regular user, type:

     ```
     smit newkey
     ```

     OR

     ```
     newkey -u username
     ```

     For a root user on a host machine, type:

     ```
     newkey -h hostname
     ```

   - Alternatively, users can establish their own public keys by using the **chkey** or **newkey** commands.
2. Create the NIS publickey map by following the instructions in *AIX 5L Version 5.3 Network Information Services (NIS and NIS+) Guide*. The corresponding NIS **publickey.byname** map resides only on the NIS servers.
3. Uncomment the following stanzas in the **/etc/rc.nfs** file:

   ```
   #if [ -x /usr/sbin/keyserv ]; then
   #  startsrc -s keyserv
   #fi
   #if [ -x /usr/lib/netsvc/yp/rpc.ypupdated -a -d /etc/yp/`domainname` ]; then
   #  startsrc -s ypupdated
   #fi
   #DIR=/etc/passwd
   #if [ -x /usr/lib/netsvc/yp/rpc.yppasswdd -a -f $DIR/passwd ]; then
   #  startsrc -s yppasswdd
   #fi
   ```

4. Start the **keyserv**, **ypupdated**, and **yppasswdd** daemons by using the **startsrc** command.

To configure secure NFS on NIS clients, start the **keyserv** daemon by using the **startsrc** command.

Exporting a file system using Secure Network File System

You can export a secure NFS using the Web-based System Manager Network application or by using one of the following procedures.

- To export a secure NFS file system using SMIT, perform the following steps:
  1. Verify that NFS is already running by running the **lssrc -g nfs** command. The output indicates that the nfsd and the rpc.mountd daemons are active.
  2. Verify that the publickey map exists and that the keyserv daemon is running. For more information, see “Configuring secure Network File System.”
  3. Run the **smit mknfsexp** fast path.
  4. Specify the appropriate values for the PATHNAME of directory to export, MODE to export directory, and EXPORT directory now, system restart or both fields. Specify yes for the Use SECURE option field.
  5. Specify any other optional characteristics, or accept the default values.
  6. Exit SMIT. If the **/etc/exports** file does not exist, it will be created.
  7. Repeat steps 3 through 6 for each directory you want to export.
- To export a secure NFS file system by using a text editor, perform the following steps:
  1. Open the **/etc/exports** file with your favorite text editor.

2. Create an entry for each directory to be exported, using the full path name of the directory. List each directory to be exported starting in the left margin. No directory should include any other directory that is already exported. See the **/etc/exports** file documentation for a description of the full syntax for entries in the **/etc/exports** file, including how to specify the secure option.
3. Save and close the **/etc/exports** file.
4. If NFS is currently running, type:

   ```
   /usr/sbin/exportfs -a
   ```

   Using the **-a** option with the **exportfs** command sends all information in the **/etc/exports** file to the kernel.

- To export an NFS file system temporarily (that is, without changing the **/etc/exports** file), type:

  ```
  exportfs -i -o secure /dirname
  ```

  where `dirname` is the name of the file system you want to export. The **exportfs -i** command specifies that the **/etc/exports** file is not to be checked for the specified directory, and all options are taken directly from the command line.

Mounting a file system using Secure Network File system

To mount a secure NFS directory explicitly, perform the following steps:

1. Verify that the NFS server has exported the directory by running the command:

   ```
   showmount -e ServerName
   ```

   where `ServerName` is the name of the NFS server. This command displays the names of the directories currently exported from the NFS server. If the directory you want to mount is not listed, export the directory from the server.
2. Establish the local mount point by using the **mkdir** command. For NFS to complete a mount successfully, a directory that acts as the mount point (or placeholder) of an NFS mount must be present. This directory should be empty. This mount point can be created like any other directory, and no special attributes are needed.
3. Verify that the publickey map exists and that the keyserv daemon is running. For more information, see “Configuring secure Network File System” on page 214.
4. Type

   ```
   mount -o secure ServerName:/remote/directory /local/directory
   ```

   where `ServerName` is the name of the NFS server, `/remote/directory` is the directory on the NFS server you want to mount, and `/local/directory` is the mount point on the NFS client.

   Note: Only the root user can mount a secure NFS.

Enterprise identity mapping

Today’s network environments are made up of a complex group of systems and applications, resulting in the need to manage multiple user registries. Dealing with multiple user registries quickly grows into a large administrative problem that affects users, administrators, and application developers. Enterprise Identity Mapping (EIM) allows administrators and application developers to address this problem.

This section describes the problems, outlines current industry approaches, and explains the EIM approach.

Managing multiple user registries

Many administrators manage networks that include different systems and servers, each with a unique way of managing users through various user registries.

In these complex networks, administrators are responsible for managing each user’s identities and passwords across multiple systems. Additionally, administrators often must synchronize these identities and passwords. Users are burdened with remembering multiple identities and passwords and with keeping them synchronized. Because user and administrator overhead in this environment is expensive, administrators often spend valuable time troubleshooting failed login attempts and resetting forgotten passwords instead of managing the enterprise.

The problem of managing multiple user registries also affects application developers who want to provide multiple-tier or heterogeneous applications. Customers have important business data spread across many different types of systems, with each system possessing its own user registries. Consequently, developers must create proprietary user registries and associated security semantics for their applications. Although this solves the problem for the application developer, it increases the overhead for users and administrators.

Current approaches to enterprise identity mapping

Several current industry approaches for solving the problem of managing multiple user registries are available, but they all provide incomplete solutions. For example, Lightweight Directory Access Protocol (LDAP) provides a distributed user registry solution. However, to use solutions such as LDAP, administrators must manage yet another user registry and security semantics or replace existing applications that are built to use those registries.

Using this type of solution, administrators must manage multiple security mechanisms for individual resources, thereby increasing administrative overhead and potentially increasing the likelihood of security exposures. When multiple mechanisms support a single resource, the chances of changing the authority through one mechanism and forgetting to change the authority for one or more of the other mechanisms is much higher. For example, a security exposure can result when a user is appropriately denied access through one interface, but allowed access through one or more other interfaces.

After completing this work, administrators find that they have not completely solved the problem. Generally, enterprises have invested too much money in current user registries and in their associated security semantics to make using this type of solution practical. Creating another user registry and associated security semantics solves the problem for the application provider, but not the problems for users or administrators.

Another solution is to use a single sign-on approach. Several products are available that allow administrators to manage files that contain all of a user’s identities and passwords. However, this approach has several weaknesses:

- It addresses only one of the problems that users face. Although it allows users to sign on to multiple systems by supplying one identity and password, the user is still required to have passwords on other systems, or the need to manage these passwords.
- It introduces a new problem by creating a security exposure because clear-text or decryptable passwords are stored in these files. Passwords should never be stored in clear-text files or be easily accessible by anyone, including administrators.
- It does not solve the problems of third-party application developers that provide heterogeneous, multiple-tier applications. They must still provide proprietary user registries for their applications.

Despite these weaknesses, some enterprises use these solutions because they provide some relief for the multiple user registry problems.

Enterprise identity mapping usage

The EIM architecture describes the relationships between individuals or entities (such as file servers and print servers) in the enterprise and the many identities that represent them within an enterprise. In addition, EIM provides a set of APIs that allow applications to ask questions about these relationships.

For example, given a person’s user identity in one user registry, you can determine which identity in another user registry represents that same person. If the user has authenticated with one identity and you can map that identity to the appropriate identity in another user registry, the user does not need to provide credentials for authentication again. You need only know which identity represents that user in another user registry. Therefore, EIM provides a generalized identity-mapping function for the enterprise.

The ability to map between a user's identities in different registries provides many benefits. Primarily, applications can have the flexibility of using one registry for authentication while using an entirely different registry for authorization. For example, an administrator could map an SAP identity to access SAP resources.

Identity mapping requires that administrators perform the following steps:

1. Create EIM identifiers that represent people or entities in their enterprise.
2. Create EIM registry definitions that describe the existing user registries in their enterprise.
3. Define the relationship between the user identities in those registries to the EIM identifiers that they created.

No code changes are required to existing registries. Mappings are not required for all identities in a user registry. EIM allows one-to-many mappings (in other words, a single user with more than one identity in a single user registry). EIM also allows many-to-one mappings (in other words, multiple users sharing a single identity in a single user registry, which although supported is not advised for security reasons). An administrator can represent any user registry of any type in EIM.

EIM does not require copying existing data to a new repository and trying to keep both copies synchronized. The only new data that EIM introduces is the relationship information. Administrators manage this data in an LDAP directory, which provides the flexibility of managing the data in one place and having replicas wherever the information is used.

For more information about Enterprise Identity Mapping, refer to the following Web sites:

- http://publib.boulder.ibm.com/eserver/
- http://www.ibm.com/servers/eserver/security/eim/

Kerberos

Kerberos is a network authentication service that provides a means of verifying the identities of principals on physically insecure networks. Kerberos provides mutual authentication, data integrity and privacy under the realistic assumption that network traffic is vulnerable to capture, examination, and substitution.

Kerberos tickets are credentials that verify your identity. There are two types of tickets: a *ticket-granting ticket* and a *service ticket*. The ticket-granting ticket is for your initial identity request. When logging into a host system, you need something that verifies your identity, such as a password or a token. After you have the ticket-granting ticket, you can then use your ticket-granting ticket to request service tickets for specific services. This two-ticket method is the called the *trusted third-party* of Kerberos. Your ticket-granting ticket authenticates you to the Kerberos server, and your service ticket is your secure introduction to the service.

The trusted third-party or intermediary in Kerberos is called the *Key Distribution Center* (KDC). The KDC issues all the Kerberos tickets to the clients.

The Kerberos database keeps a record of every principal; the record contains the name, private key, expiration date of the principal, and some administrative information about each principal. The master KDC contains the master copy of the database and passes it to slave KDCs.

This section contains the following Kerberos information:

- "Secure remote commands overview" on page 218
- "Authenticating to AIX using Kerberos" on page 220
- "KRB5A authentication load module questions and troubleshooting information" on page 226
- "Kerberos module" on page 229

Secure remote commands overview

This section provides details about secure remote commands.

Note: Beginning with Distributed Computing Environment (DCE) version 2.2, the DCE security server can return Kerberos Version 5 tickets.

Note: Beginning with AIX 5.2, all the secure remote commands (rcmds) use the Kerberos Version 5 library provided by Network Authentication Service (NAS) version 1.3. In a DCE realm, the **ftp** command uses the GSSAPI library from the **libdce.a** DCE library, and in a native realm, the **ftp** command uses the GSSAPI library from NAS version 1.3. NAS version 1.3 is located on the Expansion Pack CD. The only LPP that is required is the **krb5.client.rte** fileset.

Note: If you are migrating to AIX 5.2 and had Kerberos Version 5 or Kerberos Version 4 installed, the installation scripts prompt the user to install **krb5.client.rte**.

The secure rcmds are **rlogin**, **rcp**, **rsh**, **telnet**, and **ftp**. These commands are known collectively as the *Standard AIX* method. (This method refers to the authentication method used by AIX 4.3 and prior releases.) The additional methods provided are Kerberos Version 5 and Kerberos Version 4.

When using the Kerberos Version 5 authentication method, the client gets a Kerberos Version 5 ticket from the DCE security server or Kerberos server. The ticket is a portion of the user's current DCE or local credentials encrypted for the TCP/IP server with which they want to connect. The daemon on the TCP/IP server decrypts the ticket. This action allows the TCP/IP server to absolutely identify the user. If the DCE or local principal described in the ticket is allowed access to the operating system user's account, the connection proceeds. The secure rcmds support Kerberos clients and servers from both Kerberos Version 5 and DCE.

In addition to authenticating the client, Kerberos Version 5 forwards the current user's credentials to the TCP/IP server. If the credentials are marked as forwardable, the client sends them to the server as a Kerberos ticket-granting ticket (TGT). On the TCP/IP server side, if a user is communicating with a DCE security server, the daemon upgrades the TGT into full DCE credentials using the **k5dcecreds** command.

The **ftp** command uses a different authentication method than the other secure rcmds. It uses the GSSAPI security mechanism to pass the authentication between the **ftp** command and the **ftpd** daemon. Using the **clear**, **safe**, and **private** subcommands, the ftp client supports data encryption.

Between operating system clients and servers, the **ftp** command allows multiple byte transfers for encrypted data connections. The standards define only single byte transfers for encrypted data connections. When connected to third-party machines and using data encryption, the **ftp** command follows the single byte transfer limit.

System configuration

For all of the secure rcmds, a system-level configuration mechanism determines which authentication methods are allowed for that system. The configuration controls both outgoing and incoming connections.

The authentication configuration consists of the **libauthm.a** library and the **lsauthent** and **chauthent** commands, that provide command line access to the **get_auth_methods** and **set_auth_methods** library routines.

The authentication method defines which method is used to authenticate a user across a network. The system supports the following authentication methods:

- Kerberos Version 5 is the most common method, as it is the basis for DCE.
- Kerberos Version 4 is used only by the rlogin, rsh, and rcp secure rcmds. It is provided to support backward compatibility only on SP™ systems. A Kerberos Version 4 ticket is not upgraded to DCE credentials.

- Standard AIX is the authentication method that is used by AIX 4.3 and prior releases.

If more than one authentication method is configured and the first method fails to connect, the client attempts to authenticate using the next authentication method configured.

Authentication methods can be configured in any order. The only exception is that Standard AIX must be the final authentication method configured, because there is no fallback option. If Standard AIX is not a configured authentication method, password authentication is not attempted and any connection attempt using this method is rejected.

You can configure the system without any authentication methods. In this case, the machine refuses all connections from and to any machine using secure rcmds. Also, because Kerberos Version 4 is only supported with the **rlogin**, **rsh**, and **rcp** commands, a system configured to use only Kerberos Version 4 does not allow connections using telnet or FTP.

Kerberos Version 5 user validation

Describes how Kerberos Version 5 validates a user.

When using the Kerberos Version 5 authentication method, the TCP/IP client gets a service ticket encrypted for the TCP/IP server. When the server decrypts the ticket, it has a secure method of identifying the user (by DCE or local principal). However, the server still needs to determine if this DCE or local principal is allowed access to the local account. Mapping the DCE or local principal to the local operating system account is handled by a shared library, **libvaliduser.a**, which has a single subroutine, called kvalid_user. If a different method of mapping is preferred, the system administrator must provide an alternative for the **libvaliduser.a** library.

DCE configuration

To use the secure rcmds, two DCE principals must exist for every network interface to which they can be connected.

The two DCE principals are:

```
host/FullInterfaceName
ftp/FullInterfaceName
```

where:

FullInterfaceName
Interface name and domain name

Local configuration

To use the secure rcmds, two local principals must exist for every network interface to which they can be connected.

The two local principals are:

```
host/FullInterfaceName@Realmname
ftp/FullInterfaceName@Realmname
```

where:

FullInterfaceName
Interface name and domain name

RealmName
Name of the local Kerberos Version 5 realm

See the following sources for related information:

- The get_auth_method and set_auth_method subroutines in *AIX 5L Version 5.3 Technical Reference: Communications Volume 2*
- The chauthent command in *AIX 5L Version 5.3 Commands Reference, Volume 1*
- The lsauthent command in *AIX 5L Version 5.3 Commands Reference, Volume 3*

Authenticating to AIX using Kerberos

AIX provides both KRB5 and KRB5A Kerberos authentication load modules. Even though both modules do Kerberos authentication, the KRB5 load module performs Kerberos principal management, whereas the KRB5A load module does not.

The KRB5 load module uses the IBM Network Authentication Services' Kerberos database interface to manipulate the Kerberos identities and principals. Using the KRB5 load module, an SWsym.AIX system administrator can manage Kerberos-authenticated users and their associated Kerberos principals by using the existing SWsym.AIX user-administration commands without any change. For example, to create an SWsym.AIX user, as well as a Kerberos principal associated with that user, run the **mkuser** command.

The KRB5A load module performs only authentication. The Kerberos principal management is done separately by using Kerberos principal-management tools. The KRB5A load module is used in an environment where Kerberos principals are stored on a non-SWsym.AIX system and cannot be managed from SWsym.AIX by using the Kerberos database interface. KRB5A is intended to be used against Microsoft Windows 2000 Active Directory server where Kerberos principal management is performed using the Active Directory account management tools and APIs.

Installing and configuring the system for Kerberos integrated login using KRB5

Network Authentication Services (IBM Kerberos implementation) is shipped on the Expansion Pack.

To install the Kerberos Version 5 client package, install the **krb5.client.rte** fileset. To install the Kerberos Version 5 server package, install the **krb5.server.rte** fileset. To install the entire Kerberos Version 5 package, install the krb5 package.

To avoid namespace collisions between DCE and Kerberos commands (that is, between the **klist**, **kinit**, and **kdestroy** commands), the Kerberos commands are installed in the **/usr/krb5/bin** and the **/usr/krb5/sbin** directories. You can add these directories to your PATH definition. Otherwise, to execute the Kerberos commands, you must specify fully qualified command path names.

Network Authentication Services documentation is provided in the **krb5.doc.***lang***.pdf|html** package, where *lang* represents the supported language.

Configuring the Kerberos Version 5 KDC and kadmin servers:

Provides information on configuring the Kerberos Version 5 KDC and kadmin servers.

Note: Do not install both DCE and Kerberos server software be installed on the same physical system. If you must do so, the default operational internet port numbers must be changed for either the DCE clients and server or for the Kerberos clients and server. In either case, such a change can affect interoperability with existing DCE and Kerberos deployments in your environment. For information about coexistence of DCE and Kerberos, refer to Network Authentication Services documentation.

Note: Kerberos Version 5 is set up to reject ticket requests from any host whose clock is not within the specified maximum clock skew of the KDC. The default value for maximum clock skew is 300 seconds (five minutes). Kerberos requires that some form of time synchronization is configured between the servers and the clients. It is recommended that you use the **xntpd** or **timed** daemons for time synchronization. To use the **timed** daemon, do the following:

1. Set up the KDC server as a time server by starting the **timed** daemon, as follows:

   ```
   timed -M
   ```

2. Start the **timed** daemon on each Kerberos client.

   ```
   timed -t
   ```

   To configure the Kerberos KDC and kadmin servers, run the **mkkrb5srv** command. For example, to configure Kerberos for the `MYREALM` realm, the `sundial` server, and the `xyz.com` domain, type the following:

   ```
   mkkrb5srv -r MYREALM -s sundial.xyz.com -d xyz.com -a admin/admin
   ```

   Wait a few minutes for the **kadmind** and **krb5kdc** commands to start from **/etc/inittab**.

Running the **mkkrb5srv** command results in the following actions:

1. Creates the **/etc/krb5/krb5.conf** file. Values for realm name, Kerberos admin server, and domain name are set as specified on the command line. The **/etc/krb5/krb5.conf** file also sets the paths for the default_keytab_name, kdc, and admin_server log files.
2. Creates the **/var/krb5/krb5kdc/kdc.conf** file. The **/var/krb5/krb5kdc/kdc.conf** file sets the values for the *kdc_ports*, *kadmin_port*, *max_life*, *max_renewable_life*, *master_key_type*, and *supported_enctypes* variables. This file also sets the paths for the *database_name*, *admin_keytab*, *acl_file*, *dict_file*, and *key_stash_file* variables.
3. Creates the **/var/krb5/krb5kdc/kadm5.acl** file. Sets up the access control for admin, root, and host principals.
4. Creates the database and one admin principal. You are asked to set a Kerberos master key and to name and set the password for a Kerberos administrative principal identity. For disaster-recovery purposes, it is critical that the master key and administrative principal identity and password are securely stored away.

For more information, see "Sample runs" on page 222 and "Error messages and recovery actions" on page 222.

Configuring the Kerberos Version 5 clients:

After Kerberos installation is complete, it is not apparent to normal users that the Kerberos technology is in use and that they have ticket-granting tickets (TGTs) associated with their running processes. The login process to the operating system remains unchanged, therefore, you must configure the system to use Kerberos as the primary means of user authentication.

To configure systems to use Kerberos as the primary means of user authentication, run the **mkkrb5clnt** command with the following parameters:

```
mkkrb5clnt -c KDC -r realm -a admin -s server -d domain -A -i database -K -T
```

For example, to configure the `sundial.xyz.com` KDC with the `MYREALM` realm, `sundial.xyz.com` admin server, the `xyz.com` domain, and the `files` database, type the following:

```
mkkrb5clnt -c sundial.xyz.com -r MYREALM -s sundial.xyz.com -d xyz.com -A -i files -K -T
```

The previous example results in the following actions:

1. Creates the **/etc/krb5/krb5.conf** file. Values for realm name, Kerberos admin server, and domain name are set as specified on the command line. Also, updates the paths for **default_keytab_name**, **kdc**, and **kadmin** log files.
2. The **-i** flag configures fully integrated login. The database entered is the location where SWsym.AIX user identification information is stored. This is different than the Kerberos principal storage. The storage where Kerberos principals are stored is set during the Kerberos configuration.
3. The **-K** flag configures Kerberos as the default authentication scheme. This allows the users to become authenticated with Kerberos at login time.
4. The **-A** flag adds an entry in the Kerberos Database to make root an admin user for Kerberos.

5. The **-T** flag acquires the server admin TGT-based admin ticket.

If a system is installed that is located in a different DNS domain than the KDC, the following additional actions must be performed:

1. Edit the **/etc/krb5/krb5.conf** file and add another entry after [domain realm].
2. Map the different domain to your realm.

For example, if you want to include a client that is in the `abc.xyz.com` domain into your `MYREALM` realm , the **/etc/krb5/krb5.conf** file includes the following additional entry:

```
[domain realm]
     .abc.xyz.com = MYREALM
```

Error messages and recovery actions:

Errors that can occur when using the **mkkrb5srv** command include the following:

- If the **krb5.conf**, **kdc.conf**, or **kadm5.acl** files already exist, the **mkkrb5srv** command does not modify the values. You receive a message that the file already exists. Any of the configuration values can be changed by editing the **krb5.conf**, **kdc.conf**, or **kadm5.acl** files.
- If you mistype something and no database is created, remove the configuration files created and rerun the command.
- If there is inconsistency between the database and configuration values, remove the database from the **/var/krb5/krb5kdc/*** directory and rerun the command.
- Make sure the **kadmind** and the **krb5kdc** daemons are started on your machine. Use the **ps** command to verify that the daemons are running. If these daemons have not started, check the log file.

Errors that can occur when using the **mkkrb5clnt** command include the following:

- Incorrect values for **krb5.conf** can be fixed by editing the **/etc/krb5/krb5.conf** file.
- Incorrect values for the **-i** flag can be fixed by editing the **/usr/lib/security/methods.cfg** file.

Files created:

The **mkkrb5srv** command creates the following files:

- **/etc/krb5/krb5.conf**
- **/var/krb5/krb5kdc/kadm5.acl**
- **/var/krb5/krb5kdc/kdc.conf**

The **mkkrb5clnt** command creates the following file:

- **/etc/krb5/krb5.conf**

The **mkkrb5clnt -i** *files* option adds the following stanza to the **/usr/lib/security/methods.cfg** file:

```
KRB5:
  program =
  options =
KRB5files:
  options =
```

Sample runs:

This section provides from sample runs.

The following is an example of the **mkkrb5srv** command:

```
# mkkrb5srv -r MYREALM -s sundial.xyz.com -d xyz.com -a admin/admin
```

Output similar to the following displays:

```
  Fileset                      Level  State      Description
  ----------------------------------------------------------------------------
Path: /usr/lib/objrepos
  krb5.server.rte            1.3.0.0  COMMITTED  Network Authentication Service
                                                 Server

Path: /etc/objrepos
  krb5.server.rte            1.3.0.0  COMMITTED  Network Authentication Service
                                                 Server
The -s option is not supported.
The administration server will be the local host.
Initializing configuration...
Creating /etc/krb5/krb5.conf...
Creating /var/krb5/krb5kdc/kdc.conf...
Creating database files...
Initializing database '/var/krb5/krb5kdc/principal' for realm 'MYREALM'
master key name 'K/M@MYREALM'
You will be prompted for the database Master Password.
It is important that you NOT FORGET this password.
Enter database Master Password:
Re-enter database Master Password to verify:
WARNING: no policy specified for admin/admin@MYREALM;
  defaulting to no policy. Note that policy may be overridden by
  ACL restrictions.
Enter password for principal "admin/admin@MYREALM":
Re-enter password for principal "admin/admin@MYREALM":
Principal "admin/admin@MYREALM" created.
Creating keytable...
Creating /var/krb5/krb5kdc/kadm5.acl...
Starting krb5kdc...
krb5kdc was started successfully.
Starting kadmind...
kadmind was started successfully.
The command completed successfully.
Restarting kadmind and krb5kdc
```

The following is an example of the **mkkrb5clnt** command:

```
mkkrb5clnt -r MYREALM -c sundial.xyz.com -s sundial.xyz.com \
           -a admin/admin -d xyz.com -i files -K -T -A
```

Output similar to the following displays:

```
Initializing configuration...
Creating /etc/krb5/krb5.conf...
The command completed successfully.
Password for admin/admin@MYREALM:
Configuring fully integrated login
Authenticating as principal admin/admin with existing credentials.
WARNING: no policy specified for host/diana.xyz.com@MYREALM;
  defaulting to no policy. Note that policy may be overridden by
  ACL restrictions.
Principal "host/diana.xyz.com@MYREALM" created.

Administration credentials NOT DESTROYED.
Authenticating as principal admin/admin with existing credentials.

Administration credentials NOT DESTROYED.
Authenticating as principal admin/admin with existing credentials.
Principal "kadmin/admin@MYREALM" modified.

Administration credentials NOT DESTROYED.
Configuring Kerberos as the default authentication scheme
Making root a Kerberos administrator
Authenticating as principal admin/admin with existing credentials.
WARNING: no policy specified for root/diana.xyz.com@MYREALM;
  defaulting to no policy. Note that policy may be overridden by
  ACL restrictions.
Enter password for principal "root/diana.xyz.com@MYREALM":
Re-enter password for principal "root/diana.xyz.com@MYREALM":
Principal "root/diana.xyz.com@MYREALM" created.

Administration credentials NOT DESTROYED.
Cleaning administrator credentials and exiting.
```

Eliminating the dependency on the kadmind daemon during authentication

The KRB5 load module will fail authentication when the **kadmind** daemon is not available. This dependency to the **kadmind** daemon during authentication can be eliminated by setting the kadmind options parameter in the **methods.cfg** file.

The possible values are No or False for disabling the **kadmind** lookups and Yes or True for enabling **kadmind** lookups (the default value is Yes). When this option is set to No, the **kadmind** daemon is not contacted during authentication. Therefore, users can log into the system regardless of the status of the **kadmind** daemon (provided that the user enters the correct password when the system prompts one). However, AIX user administration commands such as **mkuser**, **chuser**, or **rmuser** will not work to administrate Kerberos integrated users if the daemon is not available (for example, either the daemon is down or the machine is not accessible).

The default value for the kadmind option parameter is Yes. This means that **kadmind** lookups will be performed during authentication. In the default case, if the daemon is not available, the authentication may take longer.

To disable the checking of the **kadmind** daemon during authentication, modify the stanzas in the **methods.cfg** file as follows:

```
KRB5:
        program = /usr/lib/security/KRB5
        options = kadmind=no

KRB5files:
        options = db=BUILTIN,auth=KRB5
```

When the **kadmind** daemon is not available, the root user will not be able to change user passwords. In a situation such as a forgotten password, you must make the **kadmind** daemon available. Also, if a user chooses to enter a Kerberos principal name at the login prompt, the primary name of the principal name will be truncated according to the AIX user name length limitation. This truncated name will be used for AIX user identification information retrieval (for example, to retrieve your home directory value).

If the **kadmind** daemon is not available (the daemon is down or not reachable), the **mkuser** command gives the following error:

```
3004-694 Error adding "krb5user": You do not have permission.
```

Additionally, the **chuser** and **lsuser** commands will manage only AIX related attributes and not Kerberos related attributes. The **rmuser** command will not delete the Kerberos principal and the **passwd** command will fail for Kerberos authenticated users.

If the network where the **kadmind** daemon resides is not accessible, response time will be delayed. Setting the kadmind option to No in the **methods.cfg** file eliminates the delays during authentication when the machine is not accessible.

When the **kadmind** daemon is down, users cannot log in if their passwords are expired. Expired passwords need to be changed. Password changes require the availability of the **kadmind** daemon. Consequently, users who require a password change or have expired passwords, will not be able to log in when the **kadmind** daemon is down.

When you set `kadmind=no` but the **kadmind** daemon is running, **lsuser** will fail to retrieve the Kerberos related attributes. However, **login**, **su**, **passwd**, **mkuser**, **chuser**, and **rmuser** commands will succeed.

Installing and configuring the system for Kerberos integrated login using KRB5A

When the **KRB5A** load module is used for authentication, a series of steps, such as creation of Kerberos principals, must be performed.

The following section explains how to authenticate an SWsym.AIX Network Authentication Service client against an Active Directory KDC.

Install the **krb5.client.rte** file set from the Expansion Pack.

Note: The KRB5A authentication load module is only supported in AIX 5.2 and later.

Configuring the AIX Kerberos Version 5 clients with a Windows 2000 active directory server:

Use the **config.krb5** command to configure an AIX Kerberos client.

Configuring the client requires Kerberos Server information. If a Windows 2000 Active Directory server is chosen as the Kerberos server, the following options can be used with the **config.krb5** command:

```
-r realm = Windows 2000 Active Directory server domain name
-d domain = Domain name of the machine hosting the Windows 2000 Active Directory server
-c KDC = Host name of the Windows 2000 Server
-s server = Host name of the Windows 2000 Server
```

1. Use the **config.krb5** command as shown in the following example:

   ```
   config.krb5 -C -r MYREALM -d xyz.com -c w2k.xyz.com -s w2k.xyz.com
   ```

2. Windows 2000 supports DES-CBC-MD5 and DES-CBC-CRC encryption types. Change the **krb5.conf** file to contain information similar to the following:

   ```
    [libdefaults]
         default_realm = MYREALM
         default_keytab_name = FILE:/etc/krb5/krb5.keytab
         default_tkt_enctypes = des-cbc-crc des-cbc-md5
         default_tgs_enctypes = des-cbc-crc des-cbc-md5
   ```

3. Add the following stanzas in the **methods.cfg** file:

   ```
   KRB5A:
     program = /usr/lib/security/KRB5A
     options = authonly
   KRB5Afiles:
     options = db=BUILTIN,auth=KRB5A
   ```

4. On a Windows 2000 Active Directory server, do the following:
   a. Use the Active Directory Management tool to create a new user account for the *krbtest* SWsym.AIX host, as follows:
      1) Select the Users folder.
      2) Use the mouse to right-click on **New**.
      3) Choose **user**.
      4) Type the name `krbtest`.

   b. Use the **Ktpass** command from the command line to create a **keytab** file and set up the account for the SWsym.AIX host. For example, to create a keytab file called **krbtest.keytab**, type:

      ```
      Ktpass -princ host/krbtest.xyz.com@MYREALM -mapuser krbtest -pass password -out krbtest.keytab
      ```

   c. Copy the keytab file to the SWsym.AIX host system.

   d. Merge the keytab file into the **/etc/krb5/krb5.keytab** file as follows:

      ```
      $ ktutil
      ktutil: rkt krbtest.keytab
      ktutil: wkt /etc/krb5/krb5.keytab
      ktutil: q
      ```

e. Create Windows 2000 domain accounts using the Active Directory user-management tools.

f. Create SWsym.AIX accounts corresponding to the Windows 2000 domain accounts so that the login process uses Kerberos authentication, as follows:

```
mkuser registry=KRB5Afiles SYSTEM=KRB5Afiles user0
```

KRB5A authentication load module questions and troubleshooting information

The following section provides answers to KRB5A authentication load module questions and troubleshooting information.

Note: The KRB5A authentication load module is only supported in AIX 5.2 and later.

- ***How do I configure an AIX Kerberos client that authenticates against an active directory server KDC?***

  Use the **config.krb5** command to configure an AIX Kerberos client. Configuring the client requires Kerberos Server information. If a Windows 2000 Active Directory server is chosen as the Kerberos server, then use the **config.krb5** command with the following options:

  -r realm
  Active Directory domain name

  -d domain
  Domain name of the machine hosting the Active Directory service

  -c KDC
  The host name of the Windows 2000 server

  -s server
  The host name of the Windows 2000 server

  Use the **config.krb5** command as shown in the following example:

  ```
  config.krb5 -C -r MYREALM -d xyz.com -c w2k.xyz.com -s w2k.xyz.com
  ```

  Windows 2000 supports DES-CBC-MD5 and DES-CBC-CRC encryption types. Change the **krb5.conf** file to contain information similar to the following:

  ```
   [libdefaults]
        default_realm = MYREALM
        default_keytab_name = FILE:/etc/krb5/krb5.keytab
        default_tkt_enctypes = des-cbc-crc des-cbc-md5
        default_tgs_enctypes = des-cbc-crc des-cbc-md5
  ```

  Add the following stanzas in the **methods.cfg** file:

  ```
  KRB5A:
    program = /usr/lib/security/KRB5A
    options = authonly
  KRB5Afiles:
    options = db=BUILTIN,auth=KRB5A
  ```

  On the Active Directory server, do the following:

  1. Use the Active Directory Management tool to create a new user account for the *krbtest* AIX host.
     - Select The Users folder.
     - Right-click with the mouse and select New.
     - Choose user.
     - Type the name *krbtest*.
  2. Use the **Ktpass** command from the command line to create a **krbtest.keytab** file and set up the account for the AIX host as follows:

     ```
     Ktpass -princ host/krbtest.xyz.com@MYREALM -mapuser krbtest -pass password \
           -out krbtest.keytab
     ```
  3. Copy the **krbtest.keytab** file to the AIX host system.

4. Merge the **krbtest.keytab** file into the **/etc/krb5/krb5.keytab** file as follows:

   ```
   $ ktutil
   ktutil: rkt krbtest.keytab
   ktutil: wkt /etc/krb5/krb5.keytab
   ktutil: q
   ```
5. Create Windows 2000 domain accounts using the Active Directory user management tools.
6. Create AIX accounts corresponding to the Windows 2000 domain accounts such that the login process will know to use Kerberos authentication, as follows:

   ```
   mkuser registry=KRB5Afiles SYSTEM=KRB5Afiles user0
   ```

- ***How do I modify an AIX configuration for Kerberos integrated login?***

  To enable Kerberos integrated login, modify the **methods.cfg** file. The compound load-module entry must be added to the **methods.cfg** file. The authentication side is KRB5A. The database side can be chosen as either BUILTIN or LDAP. BUILTIN is the standard AIX user account repository that uses ASCII files. For example, if you choose BUILTIN as the AIX user account repository, then modify the **methods.cfg** file as follows:

  ```
  Example: Local file system is chosen as the AIX user account repository.

  KRB5A:
   program = /usr/lib/security/KRB5A
   options=authonly

  KRB5Afiles:
   options = db=BUILTIN,auth=KRB5A


  Example: LDAP is chosen as the AIX user account repository.

  KRB5A:
   program = /usr/lib/security/KRB5A
   options=authonly

  LDAP:
   program = /usr/lib/security/LDAP

  KRB5ALDAP:
   options = auth=KRB5A,db=LDAP
  ```

- ***How do I create an AIX user for Kerberos integrated login with the KRB5A load module?***

  To create an AIX user for Kerberos integrated login with the KRB5A load module, use the **mkuser** command as follows:

  ```
  mkuser registry=KRB5Afiles SYSTEM=KRB5Afiles auth_domain=MYREALM foo
  ```

- ***How do I create Kerberos principals on active directory?***

  Creating Windows 2000 user accounts implicitly creates the principals. For example, if you create a user account named foo on Active Directory then the principal foo@MYREALM associated with the foo user is also created. For information on creating users on Active Directory, see the Active Directory user management documentation.

- ***How do I change the password of Kerberos authenticated user?***

  To change the password of a Kerberos authenticated user, use the **passwd** command, as follows:

  ```
  passwd -R KRB5Afiles foo
  ```

- ***How do I remove a Kerberos authenticated user?***

  To remove a Kerberos authenticated user, use the **rmuser** command. However, this only removes the user from AIX. The user must also be removed from Active Directory using the Active Directory user management tools.

  ```
  passwd -R KRB5Afiles foo
  ```

- ***How do I migrate an AIX user to a Kerberos authenticated user?***

If the user already has an account on Active Directory, then the **chuser** command converts the user into an Kerberos authenticated user, as shown in the following example:

```
chuser registry=KRB5Afiles SYSTEM=KRB5Afiles auth_domain=MYREALM foo
```

If the user does not have an account in Active Directory then create an account on Active Directory. Then use the **chuser** command. The Active Directory account may or may not have the same AIX user name. If a different name is chosen, then use the *auth_name* attribute to map to the Active Directory name. For example, to map the `chris` AIX user name to the `christopher` Active Directory user name, type the following:

```
chuser registry=KRB5Afiles SYSTEM=KRB5Afiles auth_name=christopher auth_domain=MYREALM chris
```

- ***What do I do if the password is forgotten?***

  On Active Directory, the password must be changed by the administrator. On AIX, the root user cannot set the password of a Kerberos principal.

- ***What is the purpose of the auth_name and auth_domain attributes?***

  The *auth_name* and *auth_domain* attributes are used to map AIX user names into Kerberos principal names on Active Directory. For example, if the AIX user, `chris`, has `auth_name=christopher` and `auth_domain=SOMEREALM`, the Kerberos principal name is `christopher@SOMEREALM`. The `SOMEREALM` realm name is not the same as the `MYREALM` default realm name. This allows the `chris` user to authenticate to the `SOMEREALM` realm instead of to the `MYREALM` realm.

  These attributes are optional.

- ***Can a Kerberos-authenticated user become authenticated using standard AIX authentication?***

  The answer is yes. Perform the following actions to authenticate the Kerberos-authenticated user using AIX authentication:

  1. The user sets the AIX password (**/etc/security/passwd**) using the **passwd** command, as follows:

     ```
     passwd -R files foo
     ```

  2. Change the *SYSTEM* attribute of the user, as follows:

     ```
     chuser -R KRB5Afiles SYSTEM=compat foo.
     ```

     This changes the authentication from Kerberos to crypt.

  If you want to use crypt authentication as a backup mechanism, the *SYSTEM* attribute is changed as follows:

  ```
  chuser -R KRB5Afiles SYSTEM="KRB5Afiles or compat" foo.
  ```

- ***Do I need to set up Kerberos server on AIX when using a Windows 2000 active directory server?***

  No, because users are authenticating against an Active Directory KDC, there is no need to configure the KDC on AIX. If you want to use AIX Network Authentication Services KDC as the Kerberos server instead, then the Kerberos server must be configured.

- ***What do I do if AIX does not accept my password?***

  Check that the password meets the requirements of both AIX and Kerberos. KDC must also be configured and running correctly.

- ***What do I do if I cannot log into the system?***

  If you cannot log into the system, do the following:

  – Verify that the KDC is up and running.

    - On AIX systems, type the following:

      ```
      ps -ef | grep krb5kdc
      ```

    - On Windows 2000 systems, do the following:

      1. In the Control Panel, double-click the Administrative Tools icon
      2. Double-click the Services icon.
      3. Verify that the Kerberos Key Distribution Center is in the `Started` state.

  – On AIX systems, verify that the **/etc/krb5/krb5.conf** file points to the correct KDC, and has valid parameters.

- On AIX systems, verify that client **keytab** file contains the host ticket. For example, assume you have the **/etc/krb5/krb5.keytab** default **keytab** file. Type the following:

  ```
  $ ktutil
  ktutil: rkt /etc/krb5/krb5.keytab
  ktutil: l

  slot   KVNO   Principal
  ------ ------ ------------------------------------------------------
       1      4 host/krbtest.xyz.com@MYREALM

  ktutil: q
  ```
- Verify that, if the *auth_name* and *auth_domain* attributes are set, they refer to a valid principal name on the ADS KDC.
- Verify that the *SYSTEM* attribute is set for Kerberos login (KRB5Afiles or KRB5ALDAP).
- Verify that password is not expired.

- ***How can I disable TGT verification?***

  The host/***Host_Name*** principal is used to verify a TGT. The keys for this host principal are stored in the Kerberos default **keytab** file and the **keytab** file needs to be securely transferred from the Windows 2000 Active Directory server to the client machine as explained in the Security Guide. The TGT verification could be disabled by specifying an option in the /usr/lib/security/methods.cfg file under the KRB5A stanza as follows:

  ```
  KRB5A:
          program = /usr/lib/security/KRB5A
          options = tgt_verify=no
  KRB5Afiles:
          options = db=BUILTIN,auth=KRB5A
  ```

  The possible values for **tgt_verify** are No or False for disabling, and Yes or True for enabling. By default, the TGT verification is enabled. When **tgt_verify** is set to No, TGT verification is not performed and there is no need to transfer the keys of the host principal. This eliminates the need for the **keytab** file for authentication purposes when KRB5A module is used.

Kerberos module

The Kerberos module is a kernel extension used by the NFS client and server code. It allows the NFS client and server code to process Kerberos message integrity and privacy functions without making calls to the **gss** daemon.

The Kerberos module is loaded by the **gss** daemon. The methods used are based on Network Authentication Service version 1.2, which is, in turn, based on MIT Kerberos.

The location of the Kerberos module is: **/usr/lib/drivers/krb5.ext**.

For related information, see the **gss** daemon.

Remote authentication dial-in user service server

IBM's Remote Authentication Dial-In User Service (RADIUS) is a network access protocol designed to do authentication, authorization, and accounting. It is a port-based protocol that defines the communications between Network Access Servers (NAS) and authentication and accounting servers.

A NAS operates as a client of RADIUS. Transactions between the client and the RADIUS server are authenticated through the use of a *shared secret*, which is not sent over the network. Any user passwords sent between the client and the RADIUS server are encrypted.

The client is responsible for passing user information to designated RADIUS servers and then acting on the response that is returned. RADIUS servers are responsible for receiving user connection requests, authenticating the user, and then returning all configuration information necessary for the client to deliver service to the user. A RADIUS server can act as a proxy client to other RADIUS servers when advanced proxy information is configured. RADIUS uses User Datagram Protocol (**UDP**) as the transport protocol.

The RADIUS authentication and authorization protocol is based on the IETF RFC 2865 standard. The server also provides the accounting protocol defined in RFC 2866. Other standards supported are RFC 2284 (EAP), parts of RFC 2869 and the password expiration messages of RFC 2882. All of the above RFC standards can be found at http://www.ietf.org.

Some RADIUS features are:

- Authentication, authorization and accounting (AAA)
- Password Authentication Protocol (**PAP**), Challenge Handshake Authentication Protocol (**CHAP**), and Extensible Authentication Protocol (**EAP**) authentication methods
- Proxy of packets for both authentication and authorization
- Multiple authentication databases (user information storage)
  - UNIX (AIX **/etc/password**)
  - Local database
  - LDAP Directory
- Default authorization user policy and return attributes
  - Individual user policy definition
- SMIT interface for administration
- Supports password expiration
- Multiple server support
- Vendor-specific attribute support
- NLS enabled
- IP address allocation

Installing the RADIUS server

You can install the RADIUS server using either the **installp** command or SMIT. The RADIUS software is on the AIX base media, and the image names are radius.base and bos.msg.<lang>.rte.

If you plan to use the LDAP directory as your information database to store user names and passwords, you must install the ldap.server. The **installp** software must be installed on each RADIUS server installation.

The RADIUS daemons can be started using the SRC master commands. When started, there will be multiple **radiusd** processes running:

- One process for authorization
- One process for accounting
- One process is a monitor process for the other daemons

Upon reboot, the daemons are automatically started at run level 2. To change this routine, modify the **/etc/rc.d/rc2.d/Sradiusd** file.

RADIUS authentication

Traditional authentication uses a name and a fixed password and generally takes place when the user first logs in to a machine or requests a service. RADIUS relies on an authentication database to store user IDs, passwords, and other information.

For user authentication, the server can use a local database, UNIX passwords or LDAP. Database location is configured in the server's **/etc/radius/radiusd.conf** file during setup, or by updating the file through SMIT. See "RADIUS configuration files" on page 239 for more information on RADIUS configuration files.

User databases

The RADIUS software can use different databases to store user information.

Only one of the following types of user database can be used:

- "Local"
- "UNIX"
- "LDAP" on page 232

Local:

If either the **radiusd.conf** file's **database_location** field or SMIT's Database Location entry contains the word `Local`, then the RADIUS Server will use **/etc/radius/dbdata.bin** as the location for all of the user IDs and passwords.

The local user database is flat file that contains the user ID and password information. Passwords are saved in a hashed format. Hashing is a fast addressing technique for directly accessing data in the memory space. To add, delete, or modify user passwords, run the **raddbm** command or use SMIT. When the **radiusd** daemon starts, it reads the **radiusd.conf** file and loads the user IDs and passwords into memory.

Note: The maximum user ID length is 253 characters and the maximum password length is 128 characters.

To use the local user database, select `Local` in the **Database Location** field as shown below:

```
    Configure Server

RADIUS Directory             /etc/radius
*Database Location           [Local]
Local AVL Database File Name [dbdata.bin]
Local Accounting             [ON]

Debug Level                  [3]
.
.
.
```

UNIX:

The UNIX authentication option allows RADIUS to use the local system authentication method to authenticate the user.

To use local UNIX authentication, edit the **radiusd.conf** file's **database_location** field, or select `UNIX` in SMIT's `Database Location` field. This authentication method calls the UNIX **authenticate()** application program interface (API) to authenticate a user ID and password. Passwords are saved in the same data file that UNIX uses, such as **/etc/passwords**. User IDs and passwords are created using the **mkuser** command or through SMIT.

To use the UNIX database, select `UNIX` in the **Database Location** field as shown below:

```
    Configure Server

RADIUS Directory             /etc/radius
*Database Location           [UNIX]
Local AVL Database File Name [dbdata.bin]
Local Accounting             [ON]

Debug Level                  [3]
.
.
.
```

LDAP:

RADIUS can use LDAP Version 3 to store remote user data.

RADIUS will use LDAP Version 3 API calls to access user data remotely. LDAP Version 3 access occurs if the **database_location** field in the **/etc/radiusd.conf** file is set to `LDAP` and the server name, the LDAP administrator user ID, and LDAP administrator password are configured.

AIX uses the LDAP Version 3 client libraries that are supported and packaged in the IBM Tivoli Directory Server. LDAP is a scalable protocol and the benefit of using LDAP is that user and in-process data can be located in a centralized location, easing administration of the RADIUS server. You can use the command line utility, **ldapsearch**, to view any of the RADIUS data.

Also, LDAP must be configured and administered before it can be used for RADIUS. Read the IBM Tivoli Directory Server publications at http://publib.boulder.ibm.com/tividd/td/IBMDirectoryServer5.2.html to learn more about setting up the LDAP server.

The RADIUS server provides LDAP **ldif** files to add the RADIUS schema, including object classes and attributes, to a directory, but you must set up and configure LDAP.

A separate suffix is created specifically for RADIUS to use the RADIUS LDAP objects. This suffix is a container with the name cn=aixradius, and it contains two object classes as described in “RADIUS LDAP server configuration” on page 247. You apply a RADIUS-supplied **ldif** file that creates the suffix and RADIUS schema.

When you use LDAP as the authentication database you get the following features:

1. A user database that can be seen and accessed from all RADIUS servers
2. A list of active users
3. The feature of allowing a maximum number of logins per user ID
4. An **EAP** type that can be configured per user
5. A password expiration date.

To use the LDAP database, select `LDAP` in the **Database Location** field as shown below:

```
    Configure Server

RADIUS Directory             /etc/radius
*Database Location           [LDAP]
Local AVL Database File Name [dbdata.bin]
Local Accounting             [ON]

Debug Level                  [3]
.
.
.
```

Authentication methods

RADIUS provides two standard methods for encrypting the password supplied by the user:

- Password Authentication Protocol (**PAP**)
- Challenge Handshake Authentication Protocol (**CHAP**)

Password authentication protocol:

Password Authentication Protocol (**PAP**) provides security by coding the user's password with an MD5 hash algorithm of a value that both the client and server can construct.

It works as follows:

1. In packets that have the user password, the Authentication field contains a 16 octet random number called the Request Authenticator.
2. The Request Authenticator and the client's shared secret are put into an MD5 hash. The result is a 16 octet hash.
3. The user-provided password is padded to 16 octets with nulls.
4. The hash from step 2 is XORed (Exclusive-OR) with the padded password. This is the data sent in the packet as the *user_password* attribute.
5. The RADIUS server calculates the same hash as that in Step 2.
6. This hash is XORed with the packet data from Step 4, thus recovering the password.

Challenge handshake authentication protocol:

RADIUS also supports the use of the PPP's **CHAP** for password protection.

With CHAP, the user's password is not sent across the network. Instead, an MD5 hash of the password is sent, and the RADIUS server reconstructs the hash from the user's information, including the stored password, then compares this with the value sent by the client.

Extensible authentication protocol:

The Extensible Authentication Protocol (**EAP**) is a protocol designed to support multiple authentication methods.

EAP specifies the structure of an authentication communication between a client and an authentication server, without defining the content of the authentication data. This content is defined by the specific **EAP** method that is used for authentication. Common **EAP** methods include:

MD5-Challenge
One-Time Password
Generic Token Card
Transport Layer Security (TLS)

RADIUS takes advantage of **EAP** by specifying RADIUS attributes that are used to transfer **EAP** data between the RADIUS server and its clients. This **EAP** data can then be sent by the RADIUS server directly to back-end servers that implement the various **EAP** authentication methods.

The AIX RADIUS server supports only the MD5-Challenge **EAP** method. The **EAP** method used to authenticate a user is set, at the user level, by setting a value in the user's entry in either the local database or LDAP. By default, **EAP** is turned off for each user.

RADIUS authorization

RADIUS allows authorization attributes per user as defined in the authorization policy files, **default.auth** and **default.policy**.

Authorization attributes are valid RADIUS protocol attributes that are specified in the RFC and defined in the **/etc/radius/dictionary** file. Authorization is optional and depends on how the hardware NAS or access point is configured. You must configure authorization attributes if they are needed. Authorization only happens after a successful authentication occurs.

Policies are configurable user attribute-value pairs that can control how the user accesses the network. Policies can be defined as being global to the RADIUS server, or user-specific.

Two authorization configuration files are shipped: **/etc/radius/authorization/default.auth** and **default.policy**. The **default.policy** file is used to match the incoming access request packets. The file contains attribute-value pairs that are initially blank and must be configured to the desired settings. After authentication, the policy will determine if an access accept or access reject packet is returned to the client.

Each user can also have a ***user_id*.policy** file. If a user has a unique policy file created for their specific user ID, then that files' attributes are checked first. If the attribute-value pairs in the ***user_id*.policy** file do not exactly match, then the **default.policy** file is checked. If the attribute pairs from the access request packet do not match in either file, then an access reject packet is sent. If a match is found in one or the other file, an access accept packet is sent to the client. This effectively establishes two levels of policy.

The **default.auth** file is used as the list of attribute-value pairs to return to the client once the policy has been checked. The **default.auth** file also contains attribute-value pairs that are initially blank and must be configured to the desired settings. You must edit the **default.auth** file or use SMIT to configure the desired authorization attribute settings. Each attribute that contains a value will automatically be returned to the NAS in an access accept packet.

You can also define user-specific return authorization attributes by creating a file based on the unique user name with the .auth extension, such as ***user_id*.auth**. This custom file must reside in the **/etc/radius/authorization** directory. There is a SMIT panel that allows you create and edit each user file.

Each user's authorization attributes are sent back in an access accept packet along with any default authorization attributes found in the **default.auth** file. If the values are common in the **default.auth** file and the ***user_id*.auth** file, then the user's values override the default values. This allows for some global authorization attributes (services or resources) to all users and then for more specific, per user, level of authorization.

The authorization process is as follows:

1. At daemon startup time, the default policy and authorization lists from the **/etc/radius/authorization/default.policy** file and the **default.auth** file are read into memory.
2. Authenticate the user ID and password.
3. The incoming packet is checked for attribute-value pairs.
   a. Check the custom ***user_id*.auth** file.
   b. If no match is found, then check the **default.policy** file.
   c. If no match is found, then send an access reject packet.
4. Apply the user's authorization attributes if there are any.
   a. Read the **/etc/radius/authorization/*user_id*.auth** file and the **default.auth** file, and compare the two entries.
   b. Use the entry that is in the user's file above the global entry.
5. Return the authorization attributes in an access accept packet.

RADIUS accounting server

The RADIUS accounting server is responsible for receiving accounting requests from a client and returning responses to the client indicating that it has successfully received the request and written the accounting data.

RADIUS accounting server operation

Local accounting is enabled in the **radiusd.conf** file.

When a client is configured to use RADIUS accounting, it will generate an ACCOUNTING_START packet describing the type of service being delivered and the user to whom it is being delivered at the start of service delivery. The client will send the packet to the RADIUS accounting server, which returns an acknowledgment that the packet has been received. At the end of service delivery, the client generates an ACCOUNTING_STOP packet describing the type of service that was delivered and, optionally, statistics such as elapsed time, input and output octets, or input and output packet numbers. When the ACCOUNTING_STOP packet is received by the RADIUS accounting server, it returns an acknowledgment to the accounting client that the packet has been received.

The ACCOUNTING_REQUEST, whether for START or STOP, is submitted to the RADIUS accounting server via the network. It is recommended that the client continue attempting to send the ACCOUNTING_REQUEST packet until it receives an acknowledgment. The client can also forward requests to an alternate server or servers in the event that the primary server is down or unreachable through the use of proxy configuration. See “Proxy services” on page 236 for more information.

The following is sample accounting data written to the accounting file. The actual data written is based on what the client sends to the server in the packet. Your data can be different depending on how the client is configured:

```
Thu May 27 14:43:19 2004
 NAS-IP-Address = 10.10.10.1
 NAS-Port = 1
 NAS-Port-Type = Async
 User-Name = user
 Acct-Status-Type = Start
 Acct-Authentic = RADIUS
 Service-Type = Framed-User
 Acct-Session-Id = "0000000C"
 Framed-Protocol = PPP
 Acct-Delay-Time = 0
 Timestamp = 1085686999

Thu May 27 14:45:19 2004
 NAS-IP-Address = 10.10.10.1
 NAS-Port = 1   <-- user was physically connected to port #1 on the hardware
 NAS-Port-Type = Async
 User-Name = "rod"
 Acct-Status-Type = Stop
 Acct-Authentic = RADIUS
 Service-Type = Framed-User
 Acct-Session-Id = "0000000C"   <-- note the session id's are the same
so can match up start with stops
 Framed-Protocol = PPP
 Framed-IP-Address = 10.10.10.2   <-- IP address of user user
 Acct-Terminate-Cause = User-Request  <-- user cancelled the session
 Acct-Input-Octets = 4016
 Acct-Output-Octets = 142
 Acct-Input-Packets = 35
 Acct-Output-Packets = 7
 Acct-Session-Time = 120  <-- seconds
 Acct-Delay-Time = 0
 Timestamp = 1085687119  <-- note user was only logged on for 120 seconds (2 minutes)
```

Accounting data is written in standard RADIUS format of *attribute=value* to the local **/etc/var/radius/data/accounting** file. The data written is the accounting data in the packet, with a time stamp. If the RADIUS accounting server is unable to successfully record the accounting packet, it will not send an **Accounting_Response** acknowledgment to the client and error information will be logged to the **syslog** file.

Note: Be sure the **/var** filesystem is large enough to handle all the accounting data.

Note: Third-party Perl scripts can be used to parse the data in this file. Examples of scripts that generate reports from the accounting data can be found at http://www.pgregg.com/projects/radiusreport.

Note: The accounting packets can also be proxied.

Proxy services

Proxy services allow the RADIUS server to forward requests from a NAS to another RADIUS server and then return a reply message to the NAS. Proxy services are based on a realm name.

The RADIUS server can act as both a proxy server and a back-end server simultaneously. This mechanism is applicable for both accounting and authentication packets. The concept of a realm is important to understand when dealing with proxying. Realms are identifiers that are placed before or after the values normally contained in the User-Name attribute that a RADIUS server can use to identify the server to contact to start the authentication and accounting process.

Proxy is disabled by default in the **radiusd.conf** file.

Realm example

User, *Joe*, is employed by company *XYZ* in Sacramento. The realm for this area is *SAC*. However, *Joe* is currently in New York City on a remote assignment. The realm for New York City is *NYC*. When *Joe* dials into the *NYC* realm, the User-Name passed is *SAC/Joe*. This notifies the *NYC* RADIUS realm server that this packet needs to be forwarded to the server that does the authenticating and accounting for *SAC* realm users.

Using prefixes and suffixes in the user-name attribute

How an authentication or accounting packet is routed through the realm is based on the **User-Name** attribute.

The **User-Name** attribute spells out the order of realms the packet goes through in order to route it to the final server that will do the authentication or accounting. This is accomplished by stringing realms together in the **User-Name** attribute. The actual realms that are inserted into the **User-Name** attribute, which ultimately determines the path of the packet, is a decision left up to the administrator deploying the RADIUS layout. It is possible to put the names of the realm hops in front of the **User-Name** attribute, as well as behind it. The most popular characters to delineate the different realms are the slash (/) as the prefix delineator in front of the **User-Name** attribute, and ampersand (&) as the suffix delineator behind the **User-Name** attribute. Delineators are configured in the **radiusd.conf** file. The **User-Name** attribute is parsed from left to right.

An example of a **User-Name** attribute using only the prefix method is the following:

```
 USA/TEXAS/AUSTIN/joe
```

An example of a **User-Name** attribute using only the suffix method is the following:

```
joe@USA@TEXAS@AUSTIN
```

It is possible to use both prefix and suffix methods. It is important to remember when specifying the realm hops a packet will go through that the order of hops is parsed left to right, and all prefix hops are processed before processing suffix hops. The user must be authenticated, or the accounting data written, at a single node.

The following example, using both methods, yields the same result as the previous examples:

```
USA/joe@TEXAS@AUSTIN
```

Configuring proxy services

RADIUS proxy configuration information is located in the **proxy** file in the **/etc/radius** directory.

The initial **proxy** file contains example entries. There are three fields in the proxy file: **Realm Name**, **Next Hop IP address**, and **Shared Secret**.

To configure proxy rules, select from the following::

```
    Configure Proxy Rules

List all Proxy
Add a Proxy
Change / Show Characteristics of a Proxy
Remove a Proxy
```

Select the **List all Proxy** option to read the **/etc/radius/proxy** file and display the three fields in column format. The following are the column headers:

```
realm_name   next_hop_address  shared_secret
```

Select **Add a Proxy** to display the following screen. Information is retrieved from the panel and the data is appended to the bottom of the **/etc/radius/proxy** file.

Each hop of the proxy chain uses the shared secret between the two RADIUS servers. The shared secret is contained in the **/etc/radius/proxy_file**. The shared secret should be unique per proxy hop in the chain.

See “RADIUS configuration files” on page 239 for more information about shared secret creation.

To add a proxy, select from the fields as shown below:

```
             Add a Proxy
*Realm Name                           []  (max 64 chars)
*Next Hop IP address (dotted decimal)  [xx.xx.xx.xx]
*Shared Secret                        []  (minimum 6, maximum 256 chars)
```

Selecting the **Change/Show** option displays a list of the realm names. The list is displayed in a pop-up screen and you must select a realm name.

The **Remove a Proxy** option displays a list of the realm names. The list is displayed in a pop-up screen and the user must select a realm name. After a name is selected, a verification pop-up screen is displayed before the realm is removed.

The following example is the proxy configuration information section of a **radiusd.conf** file:

```
#-----------------------------------------------------------------#
#       PROXY RADIUS Information                                  #
#                                                                 #
#                                                                 #
#   Proxy_Allow              :  ON or OFF. If ON, then the server #
#                               can proxy packets to realms it    #
#                               knows of and the following        #
#                               fields must also be configured.   #
#   Proxy_Use_Table           :  ON or OFF. If ON, then the server #
#                                can use table for faster          #
#                                processing of duplicate requests  #
#                                Can be used without proxy ON, but #
#                                it is required to be ON if        #
#                                Proxy_Use_Table is set to ON.     #
#   Proxy_Realm_name          :  This field specifies the realm    #
#                                this server services.             #
#   Proxy_Prefix_delim        :  A list of separators for parsing  #
#                                realm names added as a prefix to  #
#                                the username.  This list must be  #
#                                mutually exclusive to the Suffix  #
#                                delimiters.                       #
#   Proxy_Suffix_delim        :  A list of separators for parsing  #
#                                realm names added as a suffix to  #
#                                the username.  This list must be  #
#                                mutually exclusive to the Prefix  #
#                                delimiters.                       #
#   Proxy_Remove_Hops         :  YES or NO.  If YES then the       #
#                                will remove its realm name, the   #
#                                realm names of any previous hops  #
#                                and the realm name of the next    #
#                                server the packet will proxy to.  #
#                                                                  #
#   Proxy_Retry_count         :  The number of times to attempt    #
#                                to send the request packet.       #
#                                                                  #
#   Proxy_Time_Out            :  The number of seconds to wait     #
#                                in between send attempts.         #
#                                                                  #
#------------------------------------------------------------------#
Proxy_Allow               :   OFF
Proxy_Use_Table           :   OFF
Proxy_Realm_name          :
Proxy_Prefix_delim        :   $/
Proxy_Suffix_delim        :   @.
Proxy_Remove_Hops         :   NO
Proxy_Retry_count         :   2
Proxy_Time_Out            :   3
```

Configuring the RADIUS server

The RADIUS server daemon uses several configuration files. Server configuration information is saved in the **/etc/radius/radiusd.conf** file. The packaged server configuration file is shipped with default values.

Note: All configuration files are owned by the root user and the `security` group. The server must be restarted before any modifications to the configuration files will take effect.

The following is a sample RADIUS Configure Server SMIT panel:

```
    Configure Server
RADIUS Directory            /etc/radius
*Database Location           [UNIX]
Local AVL Database File Name [dbdata.bin]
Local Accounting             [ON]
Local Accounting Directory   []

Debug Level                  [3]
Accept Reply-Message         []
Reject Reply-Message         []
Challenge Reply-Message      []
Password Expired Reply Message []
Support Renewal of Expired Passwords [NO]
Require Message Authenticator  [NO]

*Authentication Port Number  [1812]
*Accounting Port Number      [1813]

LDAP Server Name             []
LDAP Server Port Number      [389]
LDAP Server Admin Distinguished Name     []
LDAP Server Admin Password         []
LDAP Base Distinguished Name       [cn=aixradius]
LDAP Size Limit                    [0]
LDAP Hop Limit                     [0]
LDAP wait time limit               [10]
LDAP debug level                   [ 0]

Proxy Allowed                      [OFF]
Proxy Use table                    [OFF]
Proxy Realm Name                   []
Proxy Prefix Delimiters            [$/]
Proxy Suffix Delimiters            [@.]
     NOTE: prefix & suffix are mutually exclusive
Proxy Remove Hops                  [NO]
Proxy Retry Count                  [2]
Proxy Timeout                      [30]
UNIX Check Login Restrictions      [OFF]
Enable IP Pool                     [ON]
```

RADIUS configuration files

The RADIUS daemon uses several configuration files. There are files for server setup, client descriptions, and proxy configuration. Sample versions of these files are shipped in the RADIUS package. All configuration files, except the dictionary file, can be edited through the System Management Interface Tool (SMIT).

radiusd.conf file

By default, RADIUS searches for the **radiusd.conf** file in the **/etc/radius** directory. Configuration file entries must be in the formats as shown in the file. RADIUS accepts only valid keywords and values, and uses the default if a valid keyword or value is not used. When you launch the RADIUS daemons, check the SYSLOG output for configuration parameter errors. Not all configuration errors lead to the server stopping.

This file should be appropriately read-protected and write-protected because it affects the behavior of authentication and accounting servers. Also, confidential data might exist in the file.

Important: If you edit the **radiusd.conf** file, do not change the order of the entries. SMIT panels rely on the order.

The following is an example of the radiusd.file:

```
#--------------------------------------------------------------------------#
#                CONFIGURATION FILE                                        #
#                                                                          #
#  By default RADIUS will search for radiusd.conf in the           #
#  /etc/radius directory.                                          #
#                                                                  #
#  Configuration file entries need to be in the below              #
#  formats.  RADIUS will accept only valid "Keyword : value(s)",   #
#  and will use defaults, if "Keyword : value(s)" are not          #
#  present or are in error.                                        #
#                                                                  #
#  It is important to check the syslog output when launching      #
#  the radius daemons to check for configuration parameter         #
#  errors. Once again, not all configuration errors will lead to   #
#  the server stopping.                                            #
#                                                                  #
#  Lastly, this file should be appropriately read/write protected, #
#  because it will affect the behavior of authentication and      #
#  accounting, and confidential or secretive material may          #
#  exist in this file.                                             #
#                                                                  #
#  IF YOU ARE EDITING THIS FILE, DO NOT CHANGE THE ORDER OF THE    #
#  ENTRIES IN THIS FILE. SMIT PANELS DEPEND ON THE ORDER.          #
#                                                                  #
#                                                                  #
#------------------------------------------------------------------#

#------------------------------------------------------------------#
#            Global Configuration                                  #
#                                                                  #
#   RADIUSdirectory  :  This is the base directory for the RADIUS  #
#                       daemon. The daemon will search this        #
#                       directory for further configuration files. #
#                                                                  #
#   Database_location : This is the value of where the             #
#                       authentication (user ids & passwords)      #
#                       will be stored and retrieved.              #
#                       Valid values: Local, LDAP, UNIX            #
#                       UNIX  - User defined in AIX system         #
#                       Local - Local AVL Database using raddbm    #
#                       LDAP  - Central Database                   #
#                                                                  #
#   Local_Database   :  This indicates the name of the local       #
#                       database file to be used.                  #
#                       This field must be completed if the        #
#                       Database location is Local.                #
#                                                                  #
#   Debug_Level      :  This pair sets the debug level at which    #
#                       the RADIUS server will run.  Appropriate   #
#                       values are 0,3 or 9. The default is 3.     #
#                       Output is directed to location specified   #
#                       by *.debug stanza in /etc/syslog.conf      #
#                                                                  #
#                       Each level increases the amount of messages#
#                       sent to syslog. For example "9" includes   #
#                       the new messages provided by "9" as well   #
#                       as all messages generated by level 0 and 3.#
#                                                                  #
#                       0 : provides the minimal output to the     #
#                           syslogd log.  It sends start up        #
#                           and end messages for each RADIUS       #
#                           process.  It also logs error           #
#                           conditions.                            #
#                                                                  #
#                       3 : includes general ACCESS ACCEPT, REJECT #
#                           and DISCARD messages for each packet.  #
#                           This level provides a general audit    #
#                           trail for authentication.              #
#                                                                  #
#                       9 : Maximum amount of log data. Specific   #
#                           values of attributes while a           #
#                           transaction is passing thru            #
#                           processing and more.                   #
#                           [NOT advised under normal operations]  #
#                                                                  #
#------------------------------------------------------------------#
RADIUSdirectory   : /etc/radius
Database_location : UNIX
Local_Database    : dbdata.bin
Debug_Level       : 3
#------------------------------------------------------------------#
#                 Accounting Configuration                         #
#                                                                  #
#  Local_Accounting :  When this flag is set to ON or TRUE a file  #
#                      will contain a record of ACCOUNTING START   #
#                      and STOP packets received from the Network  #
#                      Access Server(NAS).  The default log file   #
#                      is:                                         #
#                      /var/radius/data/accounting                 #
#                                                                  #
#  Local_accounting_loc  : /var/radius/data/accounting             #
#                        path and file name of the local           #
#                        accounting data file. Used only if Local_ #
#                        Accounting=ON. If the default is          #
#                        changed, then the path and file need to   #
#                        to be created (with proper permissions)   #
#                        by the admin.                             #
#                                                                  #
#------------------------------------------------------------------#
Local_Accounting     : ON
Local_Accounting_loc : /var/radius/data/accounting
#------------------------------------------------------------------#
#     Reply Message Attributes                                     #
#                                                                  #
#     Accept_Reply-Message : Sent when the RADIUS server           #
#                            replies with an Access-Accept packet  #
#                                                                  #
#     Reject_Reply-Message : Sent when the RADIUS server           #
#                            replies with an Access-Reject packet  #
#                                                                  #
#     Challenge_Reply-Message   : Sent when the RADIUS server      #
#                                 replies with an Access-Challenge #
#                                 packet                           #
#------------------------------------------------------------------#
Accept_Reply-Message :
Reject_Reply-Message :
Challenge_Reply-Message :
Password_Expired_Reply-Message :
#------------------------------------------------------------------#
#    Support Renewal of Expired Password                           #
#                                                                  #
#    Allow_Password_Renewal: YES or NO                             #
#                            Setting this attribute to YES allows  #
#                            users to update their expired password#
#                            via the RADIUS protocol. This requires#
#                            the hardware support of               #
#                            Access-Password-Request packets.      #
#------------------------------------------------------------------#
Allow_Password_Renewal  : NO
#------------------------------------------------------------------#
#    Require Message Authenticator in Access-Request               #
#                                                                  #
#    Require_Message_Authenticator: YES or NO                      #
#                                   Setting this attribute to YES  #
#                                   checks message authenticator   #
#                                   in Access-Request packet.If not#
#                                   present, it will discard the   #
#                                   packet.                        #
#------------------------------------------------------------------#
Require_Message_Authenticator : NO
#------------------------------------------------------------------#
#       Servers ( Authentication and Accounting )                  #
#                                                                  #
#   Authentication_Ports : This field indicates on which port(s)   #
#                          the authentication server(s) will listen#
#                          on.  If the field is blank an           #
#                          authentication daemon will not be       #
#                          started.                                #
#                          The value field may contain more than   #
#                          one value.  Each value is REQUIRED to   #
#                          be separated by a comma ','.            #
#                                                                  #
#                          The value field must contain a numeric  #
#                          value, like "6666".  In this case a     #
#                          server daemon will listen on "6666".    #
#                                                                  #
#   Accounting_Ports     : The same as authentication_Ports.  See  #
#                          above definitions.                      #
#                                                                  #
# [NOTE] There is no check for port conflicts.  If a server is     #
#        currently running on the specified port the deamon will   #
#        error and not run.  Be sure to check the syslog output    #
#        insure that all servers have started without incident.   #
#                                                                  #
#                                                                  #
# [Example]                                                        #
#   Authentication_Ports  : 1812,6666 (No Space between commas)    #
#                                                                  #
#   In the above example a sever will be start for each port       #
#   specified.  In the case                                        #
#                                                                  #
#           6666 : port 6666                                       #
#                                                                  #
#------------------------------------------------------------------#
Authentication_Ports  : 1812
Accounting_Ports      : 1813
#------------------------------------------------------------------#
#       LDAP Directory User Information                            #
#                                                                  #
#   Required if RADIUS is to connect to a LDAP Version 3 Directory #
#   and the Database_location field=LDAP                           #
#                                                                  #
#   LDAP_User     : User ID which has admin permission to connect  #
#                   to the remote (LDAP) database. This is the     #
#                   the LDAP administrator's DN.                   #
#                                                                  #
#   LDAP_User_Pwd : Password associated with the above User Id     #
#                   which is required to authenticate to the LDAP  #
#                   directory.                                     #
#                                                                  #
#------------------------------------------------------------------#
LDAP_User        : cn=root
LDAP_User_Pwd    :
#------------------------------------------------------------------#
#       LDAP Directory Information                                 #
#                                                                  #
#   If the Database_location field is set to "LDAP" then the       #
#   following fields need to be completed.                         #
#                                                                  #
#   LDAP_Server_name     : This field specifies the fully qualified#
#                          host name where the LDAP Version 3      #
#                          Server is located.                      #
#   LDAP_Server_Port     : The TCP port number for the LDAP server #
#                          The standard LDAP port is 389.          #
#   LDP_Base_DN          : The distinguished name for search start #
#   LDAP_Timeout         : # seconds to wait for a response from   #
#                          the LDAP server                         #
#   LDAP_Hoplimit        : maximum number of referrals to follow   #
#                          in a sequence                           #
#   LDAP_Sizelimit       : size limit (in entries) for search      #
#   LDAP_Debug_level     : 0=OFF 1=Trace ON                        #
#                                                                  #
#------------------------------------------------------------------#
LDAP_Server_name          :
LDAP_Server_port          :  389
LDAP_Base_DN              :  cn=aixradius
LDAP_Timeout              :  10
LDAP_Hoplimit             :  0
LDAP_Sizelimit            :  0
LDAP_Debug_level          :  0
#------------------------------------------------------------------#
#       PROXY RADIUS Information                                   #
#                                                                  #
#                                                                  #
#   Proxy_Allow               :  ON or OFF. If ON, then the server #
#                                can proxy packets to realms it    #
#                                knows of and the following        #
#                                fields must also be configured.   #
#   Proxy_Use_Table           :  ON or OFF. If ON, then the server #
#                                can use table for faster          #
#                                processing of duplicate requests  #
#                                Can be used without proxy ON, but #
#                                it is required to be ON if        #
#                                Proxy_Use_Table is set to ON.     #
#   Proxy_Realm_name          :  This field specifies the realm    #
#                                this server services.             #
#   Proxy_Prefix_delim        :  A list of separators for parsing  #
#                                realm names added as a prefix to  #
#                                the username.  This list must be  #
#                                mutually exclusive to the Suffix  #
#                                delimiters.                       #
#   Proxy_Suffix_delim        :  A list of separators for parsing  #
#                                realm names added as a suffix to  #
#                                the username.  This list must be  #
#                                mutually exclusive to the Prefix  #
#                                delimiters.                       #
#   Proxy_Remove_Hops         :  YES or NO.  If YES then the       #
#                                will remove its realm name, the   #
#                                realm names of any previous hops  #
#                                and the realm name of the next    #
#                                server the packet will proxy to.  #
#                                                                  #
#   Proxy_Retry_count         :  The number of times to attempt    #
#                                to send the request packet.       #
#                                                                  #
#   Proxy_Time_Out            :  The number of seconds to wait     #
#                                in between send attempts.         #
#                                                                  #
#------------------------------------------------------------------#
Proxy_Allow               :   OFF
Proxy_Use_Table           :   OFF
Proxy_Realm_name          :
Proxy_Prefix_delim        :   $/
Proxy_Suffix_delim        :   @.
Proxy_Remove_Hops         :   NO
Proxy_Retry_count         :   2
Proxy_Time_Out            :   30
#------------------------------------------------------------------#
#    Local Operating System Authentication Configuration           #
#                                                                  #
#  UNIX_Check_Login_Restrictions  : ON or OFF. If ON, during       #
#                                   local operating system authen- #
#                                   tication, a call to            #
#                                   loginrestrictions() will be    #
#                                   made to verify the user has    #
#                                   no local login restrictions.   #
#                                                                  #
#------------------------------------------------------------------#
UNIX_Check_Login_Restrictions : OFF
#------------------------------------------------------------------#
#    Global IP Pooling Flag                                        #
#                                                                  #
#  Enable_IP_Pool : ON or OFF. If ON, then RADIUS Server will do   #
#                   IP address assignment from a pool of addresses #
#                   defined to the RADIUS server.                  #
#                                                                  #
#------------------------------------------------------------------#
Enable_IP_Pool            :   OFF
#------------------------------------------------------------------#
```

/etc/radius/clients file

The **clients** file contains a list of clients that are allowed to make requests of the RADIUS server. Typically, for each client, NAS or AP, you must enter the client IP address along with the shared secret between the RADIUS server and the client and an optional *poolname* for IP pooling.

The file consists of entries in the following form:

```
<Client IP Address>    <Shared Secret>    <Pool Name>
```

A sample entry list appears as follows:

```
10.10.10.1      mysecret1      floor6
10.10.10.2      mysecret2      floor5
```

A shared secret is a character string that is configured on both the client hardware and on the RADIUS server. The maximum length of the shared secret is 256 bytes and is case sensitive. The shared secret is not sent in any of the RADIUS packets and is never sent over the network. System administrators must make sure the exact secret is configured on both sides (client and RADIUS server). The shared secret is used for encrypting the user password information and can be used for verifying message integrity by the use of a Message Authentication attribute.

Each client's shared secret should be unique in the **/etc/radius/clients** file and, like any good password, it is best to use a mixture of uppercase/lowercase letters, numbers, and symbols in the secret. To keep a shared secret secure, make it at least 16 characters in length. The **/etc/radius/clients** file can be modified using SMIT. The shared secret should be changed often to prevent dictionary attacks.

The *poolname* is the name of the pool from which global IP addresses are allocated during dynamic translation. The system administrator creates the *poolname* when setting up the RADIUS server. Using a SMIT panel, the *poolname* is added from **Configure Proxy Rules** → **IP Pool** → **Create an IP Pool**. It is used during server side IP pooling.

/etc/radius/dictionary file

The **dictionary** file contains descriptions of the attributes that are defined by the RADIUS protocol and supported by the AIX RADIUS Server. It is used by the RADIUS daemon when validating and creating packet data. Vendor-specific attributes should also be added here. The dictionary file can be modified using any editor. There is no SMIT interface.

The following is part of a sample dictionary file:

```
########################################################################
#                                                                      #
#   This file contains dictionary translations for parsing             #
#   requests and generating responses.  All transactions are           #
#   composed of Attribute/Value Pairs.  The value of each attribute    #
#   is specified as one of 4 data types.  Valid data types are:        #
#                                                                      #
#   string - 0-253 octets                                              #
#   ipaddr - 4 octets in network byte order                            #
#   integer - 32 bit value in big endian order (high byte first)       #
#   date - 32 bit value in big endian order - seconds since            #
#                                  00:00:00 GMT,  Jan.  1,  1970       #
#                                                                      #
#   Enumerated values are stored in the user file with dictionary      #
#   VALUE translations for easy administration.                        #
#                                                                      #
#   Example:                                                           #
#                                                                      #
#   ATTRIBUTE         VALUE                                            #
#   ---------------   -----                                            #
#   Framed-Protocol = PPP                                              #
#   7               = 1     (integer encoding)                         #
#                                                                      #
########################################################################
ATTRIBUTE      User-Name                     1       string
ATTRIBUTE      User-Password                 2       string
ATTRIBUTE      CHAP-Password                 3       string
ATTRIBUTE      NAS-IP-Address                4       ipaddr
ATTRIBUTE      NAS-Port                      5       integer
ATTRIBUTE      Service-Type                  6       integer
ATTRIBUTE      Framed-Protocol               7       integer
ATTRIBUTE      Framed-IP-Address             8       ipaddr
ATTRIBUTE      Framed-IP-Netmask             9       ipaddr
ATTRIBUTE      Framed-Routing                10      integer
ATTRIBUTE      Filter-Id                     11      string
.
.
.
```

Note: Any attribute that is defined in the **default.policy** file or the **default.auth** file (or for a specific **user_id.policy** or **user_id.auth** file), must be a valid RADIUS attribute as defined in the local AIX dictionary configuration file. If an attribute is not found in the dictionary, the **radiusd** daemon does not load and an error message is logged.

Note: If the dictionary, the **default.policy** file and the **default.auth** file, for the system is modified, you must restart the RADIUS daemons by running the **stopsrc** command and the **startsrc** command or by using SMIT.

/etc/radius/proxy file

The **/etc/radius/proxy** file is a configuration file that supports the proxy feature. This file is used to map known realms that the proxy server can forward packets to. The **/etc/radius/proxy** file uses the IP address of the server that handles packets for that realm and the shared secret between the two servers.

The following are fields in the **proxy** file: **Realm Name**, **Next Hop IP address**, and **Shared Secret**. The **proxy** file can be modified using SMIT.

The file looks similar to the following:

```
# @(#)91 1.3  src/rad/usr/sbin/config_files/proxy, radconfig, radius530 1/23/04 13:11:14
#######################################################################
#                                                                     #
#       This file contains a list of proxy realms which are           #
#       authorized to send/receive proxy requests/responses to/from   #
#       this RADIUS server and their Shared secret used in encryption.#
#                                                                     #
#       The first field is the name of the realm of the remote RADIUS #
#       Server.                                                       #
#                                                                     #
#       The second field is a valid IP address for the remote RADIUS  #
#       Server.                                                       #
#                                                                     #
#       The third column is the shared secret associated with this    #
#       realm.                                                        #
#                                                                     #
#       NOTE: This file contains sensative security information and   #
#             precautions should be taken to secure access to this    #
#             file.                                                   #
#                                                                     #
#######################################################################
# REALM NAME               REALM IP               SHARED SECRET
#----------------       -------------------     ----------------------
#  myRealm                10.10.10.10             sharedsec
```

The shared secret should be 16 characters in length. The same shared secret must be configured on the RADIUS server next hop.

/var/radius/data/accounting file

The **/var/radius/data/accounting** file is empty when first installed. Data is written to the file based on what the client sends in the ACCOUNTING START and ACCOUNTING STOP packets.

The following is a sample of the type of data the AIX RADIUS server writes to the **/var/radius/data/accounting** file. Your information will be different depending on how your system hardware is set up.

```
Thu May 27 14:43:19 2004
 NAS-IP-Address = 10.10.10.1
 NAS-Port = 1
 NAS-Port-Type = Async
 User-Name = "rod"
 Acct-Status-Type = Start
 Acct-Authentic = RADIUS
 Service-Type = Framed-User
 Acct-Session-Id = "0000000C"
 Framed-Protocol = PPP
 Acct-Delay-Time = 0
 Timestamp = 1085686999

Thu May 27 14:45:19 2004
 NAS-IP-Address = 10.10.10.1
 NAS-Port = 1    <-- rod was physically connected to port #1 on the hardware
 NAS-Port-Type = Async
 User-Name = "rod"
 Acct-Status-Type = Stop
 Acct-Authentic = RADIUS
 Service-Type = Framed-User
 Acct-Session-Id = "0000000C"   <-- note the session id's are the same so can match up start with stops
 Framed-Protocol = PPP
 Framed-IP-Address = 10.10.10.2   <-- IP address of user rod
 Acct-Terminate-Cause = User-Request  <-- user cancelled the session
 Acct-Input-Octets = 4016
 Acct-Output-Octets = 142
 Acct-Input-Packets = 35
 Acct-Output-Packets = 7
 Acct-Session-Time = 120  <--- seconds
 Acct-Delay-Time = 0
 Timestamp = 1085687119  <--- note "rod" was only logged on for 120 seconds (2 minutes)
```

RADIUS LDAP server configuration

When LDAP user authentication is configured, the LDAP server schema must be updated. The LDAP system administrator must add AIX RADIUS defined attributes and objectclasses to the LDAP directory before defining LDAP RADIUS users.

You must add a suffix to the LDAP server. The suffix for RADIUS is named `cn=aixradius`. A suffix is a distinguished name that identifies the top entry in a directory hierarchy.

When a suffix is added, the LDAP directory has an empty container. A *container* is an empty entry that can be used to partition the namespace. A container is similar to a file system directory, where it can have directory entries beneath it. User profile information can then be added to the LDAP directory through SMIT. The LDAP administrator ID and password are stored in the **/etc/radius/radiusd.conf** file and can be configured through SMIT on a RADIUS server.

To organize the information stored in LDAP directory entries, the schema defines object classes. An object class consists of a set of required and optional attributes. Attributes are in the form of `type=value` pairs, in which the type is defined by a unique object identifier (OID) and the value has a defined syntax. Every entry in the LDAP directory is an instance of an object.

Note: The object class, by itself, does not define a directory information tree or namespace. This only occurs when entries are created and the specific instance of object classes are given unique distinguished names. For example, when a container object class is given a unique DN, it can then be associated with two other entries which are instances of the object class organizational unit. The result is a tree-like structure or namespace.

Object classes are specific to the RADIUS server and are applied from an **ldif** file. Some of the attributes are existing LDAP schema attributes and some are specific to RADIUS. The new RADIUS object classes are structural and abstract.

For security purposes, the binds to the LDAP server use the simple bind or SASL API call, **ldap_bind_s** which will include the DN and, CRAM-MD5 as the authentication method, and the LDAP administrator password. This will transmit message digests rather than the password themselves over the network. CRAM-MD5 is a security mechanism where there is not special configuration necessary on either side (client or server).

Note: All of the attributes in the object classes are single-value.

The following LDAP schema files are located in the **/etc/radius/ldap** directory: **IBM.V3.radiusbase.schema.ldif**, and **IBM.V3.radius.schema.ldif**.

The **IBM.V3.radiusbase.schema.ldif** file defines the predefined top level object class for the RADIUS server, which is `cn=aixradius`. It also creates the following branches under the `cn=aixradius` object class:

```
1) ou=ibm-radiususer
2) ou=ibm-radiusactiveusers
```

You can add the required information by using the following command:

```
ldapadd -D ldap_admin_id -w password -i /etc/radius/ldap/IBM.V3.radiusbase.schema.ldif
```

Run this command on the LDAP server system or run remotely if the **-h** (host system name) option is added.

The **IBM.V3.radius.schema.ldif** file defines the RADIUS-specific attributes and object classes. You can add the new RADIUS attributes and object classes by typing the following command:

```
ldapmodify -D ldap_admin_id -w password -i /etc/radius/ldap/IBM.V3.radius.schema.ldif
```

You must also specify LDAP as the database location through SMIT and enter the LDAP server name and administrator password. In doing this, you are ready to add RADIUS LDAP users to the directory through SMIT.

RADIUS LDAP name space overview

At the top of the name space hierarchy is the cn=aixradius container. Below cn=aixradius, there are two organizational units (OU). OUs are containers that help make the entries unique.

RADIUS LDAP schema

The following figure graphically depicts the RADIUS LDAP schema.

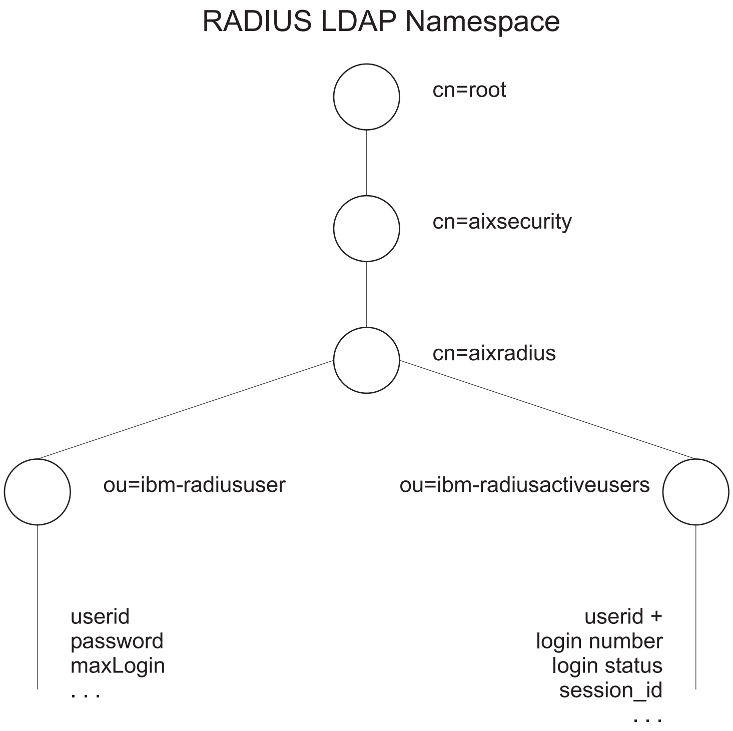


Figure 19. RADIUS LDAP Namespace

User profile object class

User profiles must be entered into the system before the RADIUS server can authenticate a user to the system. Profiles contain the user ID and password. User profile objects provide the data about the specific individuals that have access to the network and contain authentication information. The **ibm-radiusUserInstance** object class is accessed synchronously with the LDAP API calls from the daemon. The unique field, which is the start of the DN is the user ID. The **MaxLoginCount** field limits the number of times the LDAP user can log in.

Active login list object class

The active login list represents the data that contains information about the users currently logged in. There are multiple records per user with a starting record of login_number = 1, up to the MaxLoginCount number of 5. The session ID is taken from the RADIUS start_accounting message. The partially completed records are created when an **ibm-radiusUserInstance** object is created. This means that most of the fields are empty before RADIUS accounting packets are received. Once a RADIUS start_accounting message is received, the **ibm-radiusactiveusers** object is updated to specify that the user is now currently logged in, and the unique session information is written to the correct login number. After the `stop_accounting` message is received, the information in the active login list record is cleared. The active login record is updated to reflect that the user is now logged off. The session numbers in the start and stop accounting messages are the same unique number. The object class will be accessed synchronously in the LDAP API calls.

Password expiration

Password expiration allows the RADIUS client to be notified when a user's password has expired, and to update the user's password through the RADIUS protocol.

Password expiration involves supporting four more packet types and a new attribute. The new packet types are shipped in the AIX dictionary and the password expiration feature must be turned on.

It may not be desirable in every RADIUS installation to allow expired password updating through RADIUS. An entry in the **radiusd.conf** file gives you the option to allow or disallow support for expired password changing through RADIUS. The default for this option is to disallow. You can add a `Password_Expired_Reply_Message` user reply message and this is returned in the password-expired packet. Password attributes, both old and new, must be encrypted and decrypted with the PAP method.

Vendor-specific attributes

Vendor-specific attributes (VSA) are defined by remote-access server vendors, usually hardware vendors, to customize how RADIUS works on their servers.

The vendor-specific attributes are necessary if you want to give users permission for more than one type of access. The VSAs may be used in combination with RADIUS-defined attributes.

VSAs are optional, but if the NAS hardware requires additional attributes to be configured in order to function properly, you must add the VSAs to the dictionary file.

VSAs can also be used for further authorization. Along with **User-Name** and **Password**, you can use VSAs for authorization. On the server side, the user authorization policy file contains the list of attributes to be checked in the Access-Request packet for a particular user. If the packet does not contain the attributes listed in the users file, then an access_reject is sent back to NAS. VSAs can also be used as an attribute=value pair list in the ***user_id*.policy** file.

The following is a sample VSA section taken from the dictionary:

```
########################################################################
#                                                                      #
#   This section contains examples of dictionary translations for      #
#   parsing vendor specific attributes (vsa). The example below is for #
#   the vendor Cisco. Before defining an Attribute/Value pair for a    #
#   vendor a "VENDOR" definition is needed.                            #
#                                                                      #
#   Example:                                                           #
#                                                                      #
#   VENDOR          Cisco          9                                   #
#                                                                      #
#   VENDOR:  This specifies that the Attributes after this entry are   #
#            specific to Cisco.                                        #
#   Cisco :  Denotes the Vendor name                                   #
#   9     :  Vendor Id defined in the "Assigned Numbers" RFC           #
#                                                                      #
########################################################################

#VENDOR          Cisco           9

#ATTRIBUTE       Cisco-AVPair              1       string
#ATTRIBUTE       Cisco-NAS-Port            2       string
```

```
#ATTRIBUTE       Cisco-Disconnect-Cause    195     integer
#
#----------------Cisco-Disconnect-Cause---------------------------------#
#
#VALUE           Cisco-Disconnect-Cause    Unknown                 2
#VALUE           Cisco-Disconnect-Cause    CLID-Authentication-Failure 4
#VALUE           Cisco-Disconnect-Cause    No-Carrier              10
#VALUE           Cisco-Disconnect-Cause    Lost-Carrier            11
#VALUE           Cisco-Disconnect-Cause    No-Detected-Result-Codes    12
#VALUE           Cisco-Disconnect-Cause    User-Ends-Session       20
#VALUE           Cisco-Disconnect-Cause    Idle-Timeout            21
#VALUE           Cisco-Disconnect-Cause    Exit-Telnet-Session     22
#VALUE           Cisco-Disconnect-Cause    No-Remote-IP-Addr       23
```

RADIUS reply-message support

A reply-message is text that you create and configure in the **radiusd.conf** file.

It is intended for the NAS or AP to return as a string to the user. These can be a success, failure or challenge message. They are readable text fields and their contents are implementation-dependent and configured at server configuration time. The default for these attributes is no text. You may configure all, none, or one, two, or three attributes.

RADIUS supports the following Reply-Message Attributes:

- Accept Reply-Message
- Reject Reply-Message
- CHAP Reply-Message
- Password Expired Reply-Message

These attributes are added to the **radiusd.conf** configuration file and read into a global configuration structure at daemon start time. Set these values using SMIT RADIUS Panels as part of the **Configure Server** option. The maximum number of characters in each string is 256 bytes.

The function is implemented as follows:

1. When the **radiusd** daemon starts, it will read the **radiusd.conf** file and set the Reply-Message attributes.
2. When an access request packet is received, the user is authenticated.
3. If the authentication response is an access accept, then the Accept Reply-Message text is checked. If the text is present, the string is returned in the access accept packet.
4. If the authentication is rejected, then the Reject Reply-Message text is checked and returned in the access reject packet.
5. If the Authentication is challenged, then the CHAP Reply-Message attribute is checked and sent as part of the Access-Challenge packet.

RADIUS server IP pool configuration

With the RADIUS server you can assign an IP address dynamically from an IP address pool.

IP address allocation is part of the authorization process and is done after authentication. The system administrator must assign a unique IP per user. To provide the user with an IP address dynamically, the RADIUS server provides three options:

- Framed Pool Attribute
- Using the Vendor Specific Attribute
- RADIUS Server Side IP pooling

Framed Pool Attribute

The IP pool *poolname* must be defined on the Network Access Server (NAS). The NAS must be RFC2869-compliant for the RADIUS server to send an **Framed-Pool** attribute in an Access-Accept pack (type 88 attribute). The system administrator must configure the NAS and update the authorization attributes for the user by including the **Framed-Pool** attribute in either the global **default.auth** file or the **user.auth** file on the RADIUS server. The dictionary file in the RADIUS server includes this attribute:

```
ATTRIBUTE    Framed-Pool    88       string
```

If the NAS cannot use multiple address pools, the NAS ignores this attribute. The address pool on the NAS contains a list of IP addresses. The NAS dynamically picks one of the IP addresses defined in the specified pool and assigns it to the user.

Vendor Specific Attributes

Some independent software vendors (ISV) cannot use the **Framed-Pool** attribute, but do have the ability to define IP address pools. The RADIUS server can utilize these address pools by using the Vendor-Specific Attribute (VSA) model. For example, a Cisco NAS provides an attribute called Cisco-AVPair. The dictionary file in the RADIUS server includes this attribute:

```
VENDOR     Cisco          9
ATTRIBUTE  Cisco-AVPair   1       string
```

When the NAS sends an Access-Request packet, it includes this attribute with `Cisco-AVPair="ip:addr-pool=`*`poolname`*`"` where *poolname* is the name of the address pool defined on the NAS. After the request is authenticated and authorized, the RADIUS server returns the attribute in the Access-Accept packet. The NAS can then use the defined pool to allocate the IP address to the user. The system administrator must configure the NAS and update the authorization attributes for the user by including the VSA attribute in either the global **default.auth** file or the **user.auth** file on the RADIUS server.

Radius Server Side IP Pooling

The RADIUS server can be configured to generate an IP address from a pool of IP addresses. The IP address is returned in the Framed-IP-Address attribute of the Access-Accept packet.

The system administrator can define a pool of IP addresses using the SMIT interface. The addresses are maintained in the **/etc/radius/ippool_def** file. *Poolnames* are defined in the **etc/radius/clients** file. The system administrator must also configure the NAS-Port number. The RADIUS server daemon uses information from the **etc/radius/clients** and **/etc/radius/ippool_def** files to create data files. Once the daemon starts, the system administrator cannot change or add the *poolnames* or IP address ranges until the RADIUS servers have stopped. When the RADIUS server daemon is started, it reads the configuration file (**/etc/radius/radius.conf**) and if IP Allocation is enabled (`Enable_IP_Pooling=YES`), sets the global IP allocation flag (`IP_pool_flag`) to On. The daemon then checks to see if the **poolname.data** file exists. If it does, it reads the file and keeps that information in shared memory. It then updates the file and shared memory based on the requests coming in from the clients. If the file does not exist, then the daemon creates a new file using information from the **etc/radius/clients** and the **/etc/radius/ippool_def** files. The **poolname.data** file has a maximum size limit of 256 MB (AIX segment size limit). If the **poolname.data** file is more than 256 MB, the RADIUS server logs an error message and exits.

The daemon gets IP-pool details from the **/etc/radius/ippool_def** file and maintains a table of IP addresses for each pool name in shared memory. The table has entries for NAS-IP-address, NAS-port and IN USE flag. The daemon maintains a hash table that is keyed by the NAS-IP NAS-port. When requests come in from multiple users, the UDP queues the requests, and the daemon retrieves the NAS-IP and NAS-port data from the request. Using that information, it checks to see whether a *poolname* has been defined for that NAS by checking the information read from the **etc/radius/clients** file.

The daemon attempts to get an unused address from the pool. If an unused address is available, it is marked as "in use" by the NAS-IP and NAS-port flags, and is returned to the RADIUS server. The IP address is put into the **Framed-IP-Address** attribute by the daemon, and returned to the NAS in the accept packet. The **poolname.data** file is also updated to be in sync with the information in shared memory.

If the pool does not exist, or exists but does not have any more unused addresses, an error is returned to the RADIUS server. The error `Could not allocate IP address` is logged in the log file and an Access-Reject packet is sent to the NAS by the RADIUS server.

The error codes are:

- NOT_POOLED – There is no pool defined for the **nas_ip**.
- POOL_EXHAUSTED – The pool is defined for the **nas_ip**, but all of the addresses in the pool are currently in use.

When the authentication request comes from a NAS and NAS-port combination that already has an IP address allocated, the daemon returns the previous allocation to the pool, by marking the IN USE flag to Off, and clearing the NAS-IP-address and NAS-port entries in the table. It then allocates a new IP address from the pool.

The IP address is also returned to the pool when the RADIUS server receives an Accounting-Stop packet from the NAS. The Accounting-Stop packet must contain the NAS-IP-address and NAS-port entries. The daemon accesses the **ippool_mem** file for the following cases:

- The request comes in to get a new IP address. Sets the IN USE flag to true.
- An Accounting-Stop packet is received. It releases the IP address by setting the "in use" flag to false.

In each case, the shared memory system calls ensure that the data in shared memory and the **poolname.data** files are in sync. The system administer can turn IP allocation `ON` or `OFF` using the *Enable_IP_Pooling* parameter in the RADIUS server configuration file (**radiusd.conf**). This is useful in cases where the system admin has an assigned IP address in either the global **default.auth** or **user.auth** file. To use that assigned IP address, the system administrator must set `Enable_IP_Pool = NO`.

An example of an **/etc/radius/ippool_def** file created through SMIT:

Pool Name	**Start Range**	**End Range**
Floor5	192.165.1.1	192.165.1.125
Floor6	192.165.1.200	192.165.1.253

The following is an example of an **/etc/radiusclients** file created through SMIT:

NAS-IP	**Shared Secret**	**Pool Name**
1.2.3.4	Secret1	Floor5
1.2.3.5	Secret2	Floor6
1.2.3.6	Secret3	Floor5
1.2.3.7	Secret4	

In the example above for the NAS-IP-Address 1.2.3.7, the pool name is blank. In this case, IP pooling is not done for this NAS (even if the global `IP_pool_flag = True`). When the Access-Request packet comes in, the RADIUS server does the authentication and authorization. If successful, it sends the static IP address defined in the request, or from the global **default.auth** file or **user.auth** file, in the Access-Accept packet. In this case, the NAS-Port attribute is not required.

If IP pooling is True, the system administrator has also defined a static IP address as part of the global **default.auth** or **user.auth**, or as part of the Access-Request packet. The RADIUS server replaces that IP address with the IP address allocated from the defined pool name for that NAS. If all IP addresses in the pool are in use, the server logs the error (pool is full) and sends an Access-Reject packet. The server ignores any static IP address defined in the **auth** files.

If IP pooling is True and a valid pool name is defined for the NAS, when an Access-Request packet comes in from that NAS-IP, and it does not have the NAS-Port defined, the server sends a Access-Reject packet.

The following is an example of the**Floor5.data** file created by the daemon:

IP Address	NAS-IP	NAS-Port	In Use
192.165.1.1	1.2.3.4	2	1
192.165.1.2	1.2.3.4	3	0
....	....	....	....
192.165.1.124	1.2.3.6	1	1
192.165.1.125	1.2.3.6	6	1

The following is an example of the**Floor6.data** file created by the daemon:

IP Address	NAS-IP	NAS-Port	In Use
192.165.200	1.2.3.4	1	1
192.165.201	1.2.3.4	4	1
....	....	....	....
192.165.1.252	1.2.3.4	5	0
192.165.1.253	1.2.3.4	6	1

When it is necessary to release all allocated IP addresses for a specified NAS (for example, when a NAS stops), it might be necessary to release all the IP addresses from all the pools to initialize the *poolname***.data** file. The system administrator can do with the following menu actions using SMIT:

- Clear IP Pool for a Client
- Clear entire IP Pool

SMIT Panels for IP Pool

In Client Configuration, **Add a Client**, you can enter the optional **Pool Name**. The name can be a maximum of 64 characters. When the **Pool Name** is blank, IP pooling is not done and the RADIUS server assigns the IP address defined by the system administrator through the **Framed-IP-Address** authorization attribute.

When **IP Pool** is selected, the following options display:

- List all IP Pools
- Create an IP Pool
- Change/Show Characteristics of an IP Pool
- Delete an IP Pool
- Clear IP Pool for a Client
- Clear entire IP Pool

List all IP Pools: Use this option to list the **Pool Name**, **Start Range IP address** and **Stop Range IP address**.

Create an IP Pool: Use this option to add the pool name, start range, and end range. This data is appended to the bottom of the **ippool_def** file. Checks are made to ensure there are no duplicate pool names and that the IP address ranges are disjoint. This action can only be performed when the RADIUS server daemons are not running.

Change/Show Characteristics of an IP Pool: This option shows a list of the pool names in a pop-up panel. From this panel, you must select a specific pool name. When you select a pool name, a panel with the selected name displays. When you press Enter, the data for that pool name is updated in the **ippool_def** file. This action can only be performed when the RADIUS server daemons are not running.

Delete an IP Pool: Selecting this option displays a list of pool names that you can select. When you select the pool name, the **Are You Sure** pop-up panel displays to provide a confirmation before the selected pool name is deleted. The **rmippool** script is invoked to delete the selected pool name from the **ippool_def** file. This action can only be performed when the RADIUS server daemons are not running.

Clear IP Pool for a Client: This option marks the **IN-USE** entry to `0` for the IP addresses that belong to the NAS, which means that all IP addresses for this NAS are now available. This action can only be done when the RADIUS server daemons are not running.

Clear Entire IP Pool: When this option is selected, an **Are You Sure** pop-up panel displays to provide a confirmation before the entire **ippool_mem** file is cleared. This action can only be performed when the RADIUS server daemons are not running.

RADIUS SMIT panels

When using SMIT to configure the RADIUS server, fields marked with an asterisk (*) are required fields.

The SMIT fast-path is:

```
smitty radius
```

The RADIUS Main Menu is as follows:

```
      RADIUS Server

Configure Server
Configure Clients
Configure Users
Configure Proxy Rules
Advanced Server Configuration
Start RADIUS Server daemons
Stop RADIUS Server daemons
```

The following screen capture shows a sample RADIUS Configure Server SMIT panel:

```
    Configure Server

RADIUS Directory            /etc/radius
*Database Location           [UNIX]
Local AVL Database File Name [dbdata.bin]
Local Accounting             [ON]
Local Accounting Directory   []

Debug Level                  [3]
Accept Reply-Message         []
Reject Reply-Message         []
Challenge Reply-Message      []
Password Expired Reply Message []
Support Renewal of Expired Passwords [NO]
Require Message Authenticator  [NO]

*Authentication Port Number  [1812]
*Accounting Port Number      [1813]

LDAP Server Name             []
LDAP Server Port Number      [389]
LDAP Server Admin Distinguished Name     []
LDAP Server Admin Password         []
LDAP Base Distinguished Name       [cn=aixradius]
LDAP Size Limit                    [0]
LDAP Hop Limit                     [0]
LDAP wait time limit               [10]
LDAP debug level                   [ 0]

Proxy Allowed                      [OFF]
Proxy Use table                    [OFF]
Proxy Realm Name                   []
Proxy Prefix Delimiters            [$/]
Proxy Suffix Delimiters            [@.]
     NOTE: prefix & suffix are mutually exclusive
Proxy Remove Hops                  [NO]
Proxy Retry Count                  [2]
Proxy Timeout                      [30]
UNIX Check Login Restrictions      [OFF]
Enable IP Pool                     [ON]
```

Detailed SMIT help information is available for all fields and menu options by pressing the **F1** key.

Random number generator

Random numbers are required when generating the Authenticator field of a RADIUS packet.

It is important to provide the best possible generator because an intruder could try to trick the RADIUS server into responding to a predicted request and then use the response to masquerade as that RADIUS server to a future access-request. The AIX RADIUS Server uses the **/dev/urandom** kernel extension to generate pseudo random numbers. This kernel extension collects entropy samples from hardware sources by way of the pseudo device driver. This device has been through NIST testing to ensure proper randomness.

Logging utilities

The RADIUS server uses SYSLOG to log activity and error information.

There are three levels of log information:

0 Only problems or errors are logged plus the starting of daemons.

3 Logs an audit trail of `access_accept`, `access_reject*`, `discard`, and `error` messages.

9 Includes 0 and 3 level logging information and much more. Only run level 9 logging to debug.

*discard messages are logged when an incoming packet is invalid and a response packet is not generated.

SYSLOG output for audit logging

The default level of logging is level 3. Logging at level 3 is used to improve the level of auditing of the RADIUS server. Depending on what level the server is logging, you can use the activities stored in the log to check for suspicious patterns of activity. If security is violated, the SYSLOG output can be used to determine how and when the violation occurred and perhaps the amount of access gained. This information is useful in the development of better security measures to prevent future problems.

For information on configuring RADIUS to use the **syslogd** daemon, see “Logging facilities” on page 178.

Viewing and understanding SYSLOG output

A Debug_Level of 0, 3, or 9 is set in the **radiusd.conf** file. The default is 3. Sample output of the debug section of the **radiusd.conf** file looks similar to the following:

```
#.
#.
#.
#   Debug_Level      :  This pair sets the debug level at which    #
#                       the RADIUS server will run.  Appropriate   #
#                       values are 0,3 or 9. The default is 3.     #
#                       Output is directed to location specified   #
#                       by *.debug stanza in /etc/syslog.conf      #
#                                                                  #
#                       Each level increases the amount of messages#
#                       sent to syslog. For example "9" includes   #
#                       the new messages provided by "9" as well   #
#                       as all messages generated by level 0 and 3.#
#                                                                  #
#                       0 : provides the minimal output to the     #
#                           syslogd log.  It sends start up        #
#                           and end messages for each RADIUS       #
#                           process.  It also logs error           #
#                           conditions.                            #
#                                                                  #
#                       3 : includes general ACCESS ACCEPT, REJECT #
#                           and DISCARD messages for each packet.  #
#                           This level provides a general audit    #
#                           trail for authentication.              #
#                                                                  #
#                       9 : Maximum amount of log data. Specific   #
#                           values of attributes while a           #
#                           transaction is passing thru            #
#                           processing and more.                   #
#                           [NOT advised under normal operations]  #
#                                                                  #
#------------------------------------------------------------------#
```

The following are examples of SYSLOG output at the 3 different debugging levels.

Accounting packet with debug level 3

```
Aug 18 10:23:57 server1 syslog: [0]:Monitor process [389288] has started
Aug 18 10:23:57 server1 radiusd[389288]: [0]:Local database (AVL) built.
Aug 18 10:23:57 server1 radiusd[389288]: [0]:Authentication process started : Pid= 549082 Port = 1812
Aug 18 10:23:57 server1 radiusd[389288]: [0]:Accounting process started : Pid= 643188 Port = 1813
Aug 18 10:23:57 server1 radiusd[643188]: [0]:Socket created [15]
Aug 18 10:23:57 server1 radiusd[643188]: [0]:Bound Accounting socket [15]
Aug 18 10:23:57 server1 radiusd[549082]: [0]:Socket created [15]
Aug 18 10:23:57 server1 radiusd[549082]: [0]:Bound Authentication socket [15]
Aug 18 10:24:07 server1 radiusd[643188]: [1]:*** Start Process_Packet() ***
Aug 18 10:24:07 server1 radiusd[643188]: [1]:Code 4, ID = 96, Port = 41639 Host = 10.10.10.10
Aug 18 10:24:07 server1 radiusd[643188]: [1]:ACCOUNTING-START - sending Accounting Ack to User [ user_id1 ]
Aug 18 10:24:07 server1 radiusd[643188]: [1]:Sending Accounting Ack of id 96 to 10.10.10.10 (client1.ibm.com)
Aug 18 10:24:07 server1 radiusd[643188]: [1]:send_acct_reply() Outgoing Packet:
Aug 18 10:24:07 server1 radiusd[643188]: [1]: Code = 5, Id = 96, Length = 20
Aug 18 10:24:07 server1 radiusd[643188]: [1]:*** Leave Process_Packet() ***
Aug 18 10:24:13 server1 radiusd[643188]: [2]:*** Start Process_Packet() ***
Aug 18 10:24:13 server1 radiusd[643188]: [2]:Code 4, ID = 97, Port = 41639 Host = 10.10.10.10
Aug 18 10:24:13 server1 radiusd[643188]: [2]:ACCOUNTING-STOP - sending Accounting Ack to User [ user_id1 ]
Aug 18 10:24:14 server1 radiusd[643188]: [2]:Sending Accounting Ack of id 97 to 10.10.10.10 (client1.ibm.com)
Aug 18 10:24:14 server1 radiusd[643188]: [2]:send_acct_reply() Outgoing Packet:
Aug 18 10:24:14 server1 radiusd[643188]: [2]: Code = 5, Id = 97, Length = 20
Aug 18 10:24:14 server1 radiusd[643188]: [2]:*** Leave Process_Packet() **
```

Accounting packets at level 9

```
Aug 18 10:21:18 server1 syslog: [0]:Monitor process [643170] has started
Aug 18 10:21:18 server1 radiusd[643170]: [0]:Local database (AVL) built.
Aug 18 10:21:18 server1 radiusd[643170]: [0]:Authentication process started : Pid= 389284 Port = 1812
Aug 18 10:21:18 server1 radiusd[643170]: [0]:Accounting process started : Pid= 549078 Port = 1813
Aug 18 10:22:03 server1 radiusd[643170]: [0]:PID = [389284] dead
Aug 18 10:22:03 server1 radiusd[643170]: [0]:PID = [549078] dead
Aug 18 10:22:03 server1 radiusd[643170]: [0]:All child processes stopped. radiusd parent stopping
Aug 18 10:22:09 server1 syslog: [0]:Monitor process [1081472] has started
Aug 18 10:22:09 server1 radiusd[1081472]: [0]:Local database (AVL) built.
Aug 18 10:22:09 server1 radiusd[1081472]: [0]:Inside client_init()
Aug 18 10:22:09 server1 radiusd[1081472]: [0]:Number of client entries read: 1
Aug 18 10:22:09 server1 radiusd[1081472]: [0]:Inside read_authorize_policy routine for file:
  /etc/radius/authorization/default.policy.
Aug 18 10:22:09 server1 radiusd[1081472]: [0]:Inside read_authorize_file routine for file:
  /etc/radius/authorization/default.policy.
Aug 18 10:22:09 server1 radiusd[1081472]: [0]:read_authorize_file() routine complete.
Aug 18 10:22:09 server1 radiusd[1081472]: [0]:Inside read_authorize_file routine for file:
  /etc/radius/authorization/default.auth.
Aug 18 10:22:09 server1 radiusd[1081472]: [0]:read_authorize_file() routine complete.
Aug 18 10:22:09 server1 radiusd[549080]: [0]:connect_to_LDAP_server:Database Location (where the data
  resides)=LDAP.
Aug 18 10:22:09 server1 radiusd[549080]: [0]:connect_to_LDAP_server:LDAP Server name= server1.austin.ibm.com.
Aug 18 10:22:09 server1 radiusd[549080]: [0]:connect_to_LDAP_server:LDAP Server port= 389.
Aug 18 10:22:09 server1 radiusd[1081472]: [0]:Authentication process started : Pid= 549080 Port = 1812
Aug 18 10:22:09 server1 radiusd[389286]: [0]:connect_to_LDAP_server:Database Location (where the data
 resides)=LDAP.
Aug 18 10:22:09 server1 radiusd[389286]: [0]:connect_to_LDAP_server:LDAP Server name= server1.austin.ibm.com.
Aug 18 10:22:09 server1 radiusd[389286]: [0]:connect_to_LDAP_server:LDAP Server port= 389.
Aug 18 10:22:09 server1 radiusd[1081472]: [0]:Accounting process started : Pid= 389286 Port = 1813
Aug 18 10:22:10 server1 radiusd[549080]: [0]:Socket created [15]
Aug 18 10:22:10 server1 radiusd[549080]: [0]:Bound Authentication socket [15]
Aug 18 10:22:10 server1 radiusd[389286]: [0]:Socket created [15]
Aug 18 10:22:10 server1 radiusd[389286]: [0]:Bound Accounting socket [15]
Aug 18 10:22:15 server1 radiusd[389286]: [1]:*** Start Process_Packet() ***
Aug 18 10:22:15 server1 radiusd[389286]: [1]:Incoming Packet:
Aug 18 10:22:15 server1 radiusd[389286]: [1]:  Code = 4, Id = 94, Length = 80
Aug 18 10:22:15 server1 radiusd[389286]: [1]:  Authenticator = 0xC5DBDDFE6EFFFDBD6AE64CA35947DD0F
Aug 18 10:22:15 server1 radiusd[389286]: [1]:    Type =  40, Length =   6, Value = 0x00000001
Aug 18 10:22:15 server1 radiusd[389286]: [1]:    Type =   1, Length =   8, Value = 0x67656E747931
Aug 18 10:22:15 server1 radiusd[389286]: [1]:    Type =   4, Length =   6, Value = 0x00000000
Aug 18 10:22:15 server1 radiusd[389286]: [1]:    Type =   8, Length =   6, Value = 0x0A0A0A01
Aug 18 10:22:15 server1 radiusd[389286]: [1]:    Type =  44, Length =   8, Value = 0x303030303062
Aug 18 10:22:15 server1 radiusd[389286]: [1]:    Type =  30, Length =  10, Value = 0x3132332D34353638
Aug 18 10:22:15 server1 radiusd[389286]: [1]:    Type =  31, Length =  10, Value = 0x3435362D31323335
Aug 18 10:22:15 server1 radiusd[389286]: [1]:    Type =  85, Length =   6, Value = 0x00000259
Aug 18 10:22:15 server1 radiusd[389286]: [1]:Starting parse_packet()
Aug 18 10:22:15 server1 radiusd[389286]: [1]:Code 4, ID = 94, Port = 41639 Host = 10.10.10.10
Aug 18 10:22:15 server1 radiusd[389286]: [1]:Acct-Status-Type = Sta
```

Level 0 authentication packet:

```
Aug 18 10:06:11 server1 syslog: [0]:Monitor process [1081460] has started
Aug 18 10:06:11 server1 radiusd[1081460]: [0]:Local database (AVL) built.
Aug 18 10:06:11 server1 radiusd[1081460]: [0]:Authentication process started : Pid= 549076 Port = 1812
Aug 18 10:06:11 server1 radiusd[1081460]: [0]:Accounting process started : Pid= 389282 Port = 18
```

Level 3 authentication packet:

```
Aug 18 10:01:32 server2 radiusd[389276]: [3]:*** Start Process_Packet() ***
Aug 18 10:01:32 server2 radiusd[389276]: [3]:Code 1, ID = 72, Port = 41638 Host = 10.10.10.10
Aug 18 10:01:32 server2 radiusd[389276]: [3]:authenticate_password_PAP: Passwords do not match, user is rejected
Aug 18 10:01:32 server2 radiusd[389276]: [3]:Authentication failed for user [user_id1] using IP [10.10.10.10]
Aug 18 10:01:32 server2 radiusd[389276]: [3]:ACCESS-REJECT - sending reject for id 72 to 10.10.10.10
  (client1.ibm.com)
Aug 18 10:01:32 server2 radiusd[389276]: [3]:send_reject() Outgoing Packet:
Aug 18 10:01:32 server2 radiusd[389276]: [3]: Code = 3, Id = 72, Length = 30
Aug 18 10:01:32 server2 radiusd[389276]: [3]:*** Leave Process_Packet() ***
Aug 18 10:01:53 server2 radiusd[389276]: [4]:*** Start Process_Packet() ***
Aug 18 10:01:53 server2 radiusd[389276]: [4]:Code 1, ID = 74, Port = 41638 Host = 10.10.10.10
Aug 18 10:01:53 server2 radiusd[389276]: [4]:authenticate_password_PAP: Passwords Match, user is authenticated
Aug 18 10:01:53 server2 radiusd[389276]: [4]:Authentication successful for user [user_id1] using IP [10.10.10.10]
Aug 18 10:01:53 server2 radiusd[389276]: [4]:Authorization successful for user [user_id1] using IP [10.10.10.10]
Aug 18 10:01:53 server2 radiusd[389276]: [4]:ACCESS-ACCEPT - sending accept for id 74 to 10.10.10.10
  (client1.ibm.com)
Aug 18 10:01:53 server2 radiusd[389276]: [4]:send_accept() Outgoing Packet:
Aug 18 10:01:53 server2 radiusd[389276]: [4]: Code = 2, Id = 74, Length = 31
Aug 18 10:01:53 server2 radiusd[389276]: [4]:*** Leave Process_Packet() **
```

Level 9 authentication packet:

```
Aug 18 10:03:56 server1 radiusd[389278]: [1]:*** Start Process_Packet() ***
Aug 18 10:03:56 server1 radiusd[389278]: [1]:Incoming Packet:
Aug 18 10:03:56 server1 radiusd[389278]: [1]:  Code = 1, Id = 77, Length = 58
Aug 18 10:03:56 server1 radiusd[389278]: [1]:  Authenticator = 0xE6CB0F9C22BB4E799854E734104FB2D5
Aug 18 10:03:56 server1 radiusd[389278]: [1]:    Type =   1, Length =   8, Value = 0x67656E747931
Aug 18 10:03:56 server1 radiusd[389278]: [1]:    Type =   4, Length =   6, Value = 0x00000000
Aug 18 10:03:56 server1 radiusd[389278]: [1]:    Type =   2, Length =  18, Value = 0x**********
**********************
Aug 18 10:03:56 server1 radiusd[389278]: [1]:    Type =   7, Length =   6, Value = 0x00000001
Aug 18 10:03:56 server1 radiusd[389278]: [1]:Starting parse_packet()
Aug 18 10:03:56 server1 radiusd[389278]: [1]:Code 1, ID = 77, Port = 41638 Host = 10.10.10.10
Aug 18 10:03:56 server1 radiusd[389278]: [1]:User-Name = "user_id1"
Aug 18 10:03:56 server1 radiusd[389278]: [1]:NAS-IP-Address = 10.10.10.10
Aug 18 10:03:56 server1 radiusd[389278]: [1]:Framed-Protocol = PPP
Aug 18 10:03:56 server1 radiusd[389278]: [1]:Leaving parse_packet()
Aug 18 10:03:56 server1 radiusd[389278]: [1]:Verifying Message-Authenticator
Aug 18 10:03:56 server1 radiusd[389278]: [1]:Message-Authenticator successfully verified
Aug 18 10:03:56 server1 radiusd[389278]: [1]:Inside proxy_request_needed() function
Aug 18 10:03:56 server1 radiusd[389278]: [1]:Proxy is not turned on
Aug 18 10:03:56 server1 radiusd[389278]: [1]:Username = [user_id1]
Aug 18 10:03:56 server1 radiusd[389278]: [1]:Client IP = [10.10.10.10]
Aug 18 10:03:56 server1 radiusd[389278]: [1]:Inside parse_for_login( user_id1 )
Aug 18 10:03:56 server1 radiusd[389278]: [1]:User_id remaining after prefix removal = [user_id1]
Aug 18 10:03:56 server1 radiusd[389278]: [1]:User_id remaining after suffix removal = [user_id1]
Aug 18 10:03:56 server1 radiusd[389278]: [1]:Inside rad_authenticate() function
Aug 18 10:03:56 server1 radiusd[389278]: [1]:Authentication request received for [client1.austin.ibm.com]
Aug 18 10:03:56 server1 radiusd[389278]: [1]:Calling get_ldap_user() to get LDAP user data
Aug 18 10:03:56 server1 radiusd[389278]: [1]:get_ldap_user:LDAP user id: user_id1.
Aug 18 10:03:56 server1 radiusd[389278]: [1]:get_ldap_user:LDAP max_login_cnt:2.
Aug 18 10:03:56 server1 radiusd[389278]: [1]:get_ldap_user:LDAP EAP_type: 4.
Aug 18 10:03:56 server1 radiusd[389278]: [1]:get_ldap_user:LDAP passwordexpiredweeks: 9.
Aug 18 10:03:56 server1 radiusd[389278]: [1]:get_ldap_active_sessions:number of free entries= 2.
Aug 18 10:03:56 server1 radiusd[389278]: [1]:get_ldap_active_session:dn retrieved=
radiusuniqueidentifier=user_id11,ou=radiusActiveUsers,cn=aixradius.
Aug 18 10:03:56 server1 radiusd[389278]: [1]:Inside get_client_secret routine for ip:10.10.10.10
Aug 18 10:03:56 server1 radiusd[389278]: [1]:Found NAS-IP = [10.10.10.10]
Aug 18 10:03:56 server1 radiusd[389278]: [1]:Found shared secret.
Aug 18 10:03:56 server1 radiusd[389278]: [1]:authenticate_password_PAP: Passwords Match, user is authenticated
Aug 18 10:03:56 server1 radiusd[389278]: [1]:is_ldap_pw:password for user has NOT expired
Aug 18 10:03:56 server1 radiusd[389278]: [1]:Authentication successful for user [user_id1] using IP [10.10.10.10]
Aug 18 10:03:56 server1 radiusd[389278]: [1]:Inside rad_authorize() routine.
Aug 18 10:03:56 server1 radiusd[389278]: [1]:Inside read_authorize_policy routine for file:
  /etc/radius/authorization/user_id1.policy.
Aug 18 10:03:56 server1 radiusd[389278]: [1]:Inside read_authorize_file routine for file:
  /etc/radius/authorization/user_id1.policy.
Aug 18 10:03:56 server1 radiusd[389278]: [1]:Did not open /etc/radius/authorization/user_id1.policy file.
  File may not be found.
Aug 18 10:03:56 server1 radiusd[389278]: [1]:Error reading policy file: /etc/radius/authorization/user_id1.policy.
Aug 18 10:03:56 server1 radiusd[389278]: [1]:rad_authorize:default policy list and userpolicy list were empty.
Aug 18 10:03:56 server1 radiusd[389278]: [1]:In create_def_copy() routine.
Aug 18 10:03:56 server1 radiusd[389278]: [1]:Successfully made a copy of the master authorization list.
Aug 18 10:03:56 server1 radiusd[389278]: [1]:Inside read_authorize_file routine for file:
  /etc/radius/authorization/user_id1.auth.
Aug 18 10:03:56 server1 radiusd[389278]: [1]:Did not open /etc/radius/authorization/user_id1.auth file.
  File may not be found.
Aug 18 10:03:56 server1 radiusd[389278]: [1]:rad_authorize:copy authorization list and user list were empty.
Aug 18 10:03:56 server1 radiusd[389278]: [1]:Authorization successful for user [user_id1] using IP [10.10.10.10]
Aug 18 10:03:56 server1 radiusd[389278]: [1]:ACCESS-ACCEPT - sending accept for id 77 to 10.10.10.10
  (client1.austin.ibm.com)
Aug 18 10:03:56 server1 radiusd[389278]: [1]:Inside proxy_response_needed() function
Aug 18 10:03:56 server1 radiusd[389278]: [1]:Proxy is not turned on
Aug 18 10:03:56 server1 radiusd[389278]: [1]:Inside get_client_secret routine for ip:10.10.10.10
Aug 18 10:03:56 server1 radiusd[389278]: [1]:Found NAS-IP = [10.10.10.10]
Aug 18 10:03:56 server1 radiusd[389278]: [1]:Found shared secret.
Aug 18 10:03:56 server1 radiusd[389278]: [1]:send_accept() Outgoing Packet:
Aug 18 10:03:56 server1 radiusd[389278]: [1]: Code = 2, Id = 77, Length = 31
Aug 18 10:03:56 server1 radiusd[389278]: [1]:send_accept() Outgoing Packet:
Aug 18 10:03:56 server1 radiusd[389278]: [1]:  Code = 2, Id = 77, Length = 31
Aug 18 10:03:56 server1 radiusd[389278]: [1]:  Authenticator = 0xCCB2B645BBEE86F5E4FC5BE24E904B2A
Aug 18 10:03:56 server1 radiusd[389278]: [1]:    Type =  18, Length =  11, Value = 0x476F6F646E65737321
Aug 18 10:03:56 server1 radiusd[389278]: [1]:*** Leave Process_Packet() ***
Aug 18 10:04:18 server1 radiusd[389278]: [2]:*** Start Process_Packet() ***
Aug 18 10:04:18 server1 radiusd[389278]: [2]:Incoming Packet:
Aug 18 10:04:18 server1 radiusd[389278]: [2]:  Code = 1, Id = 79, Length = 58
Aug 18 10:04:18 server1 radiusd[389278]: [2]:  Authenticator = 0x774298A2B6DD90D7C33B3C10C4787D41
Aug 18 10:04:18 server1 radiusd[389278]: [2]:    Type =   1, Length =   8, Value = 0x67656E747931
Aug 18 10:04:18 server1 radiusd[389278]: [2]:    Type =   4, Length =   6, Value = 0x00000000
Aug 18 10:04:18 server1 radiusd[389278]: [2]:    Type =   2, Length =  18, Value = 0x*******
*************************
Aug 18 10:04:18 server1 radiusd[389278]: [2]:    Type =   7, Length =   6, Value = 0x00000001
Aug 18 10:04:18 server1 radiusd[389278]: [2]:Starting parse_packet()
Aug 18 10:04:18 server1 radiusd[389278]: [2]:Code 1, ID = 79, Port = 41638 Host = 10.10.10.10
Aug 18 10:04:18 server1 radiusd[389278]: [2]:User-Name = "user_id1"
Aug 18 10:04:18 server1 radiusd[389278]: [2]:NAS-IP-Address = 10.10.10.10
Aug 18 10:04:18 server1 radiusd[389278]: [2]:Framed-Protocol = PPP
Aug 18 10:04:18 server1 radiusd[389278]: [2]:Leaving parse_packet()
Aug 18 10:04:18 server1 radiusd[389278]: [2]:Verifying Message-Authenticator
Aug 18 10:04:18 server1 radiusd[389278]: [2]:Message-Authenticator successfully verified
Aug 18 10:04:18 server1 radiusd[389278]: [2]:Inside proxy_request_needed() function
Aug 18 10:04:18 server1 radiusd[389278]: [2]:Proxy is not turned on
Aug 18 10:04:18 server1 radiusd[389278]: [2]:Username = [user_id1]
Aug 18 10:04:18 server1 radiusd[389278]: [2]:Client IP = [10.10.10.10]
Aug 18 10:04:18 server1 radiusd[389278]: [2]:Inside parse_for_login( user_id1 )
Aug 18 10:04:18 server1 radiusd[389278]: [2]:User_id remaining after prefix removal = [user_id1]
Aug 18 10:04:18 server1 radiusd[389278]: [2]:User_id remaining after suffix removal = [user_id1]
Aug 18 10:04:18 server1 radiusd[389278]: [2]:Inside rad_authenticate() function
Aug 18 10:04:18 server1 radiusd[389278]: [2]:Authentication request received for [client1.austin.ibm.com]
Aug 18 10:04:18 server1 radiusd[389278]: [2]:Calling get_ldap_user() to get LDAP user data
Aug 18 10:04:18 server1 radiusd[389278]: [2]:get_ldap_user:LDAP user id: user_id1.
Aug 18 10:04:18 server1 radiusd[389278]: [2]:get_ldap_user:LDAP max_login_cnt:2.
Aug 18 10:04:18 server1 radiusd[389278]: [2]:get_ldap_user:LDAP EAP_type: 4.
Aug 18 10:04:18 server1 radiusd[389278]: [2]:get_ldap_user:LDAP passwordexpiredweeks: 9.
Aug 18 10:04:18 server1 radiusd[389278]: [2]:get_ldap_active_sessions:number of free entries= 2.
Aug 18 10:04:18 server1 radiusd[389278]: [2]:get_ldap_active_session:dn retrieved=
  radiusuniqueidentifier=user_id11, ou=radiusActiveUsers, cn=aixradius.
Aug 18 10:04:18 server1 radiusd[389278]: [2]:Inside get_client_secret routine for ip:10.10.10.10
Aug 18 10:04:18 server1 radiusd[389278]: [2]:Found NAS-IP = [10.10.10.10]
Aug 18 10:04:18 server1 radiusd[389278]: [2]:Found shared secret.
Aug 18 10:04:18 server1 radiusd[389278]: [2]:authenticate_password_PAP: Passwords do not match, user is rejected
Aug 18 10:04:18 server1 radiusd[389278]: [2]:Authentication failed for user [user_id1] using IP [10.10.10.10]
Aug 18 10:04:18 server1 radiusd[389278]: [2]:ACCESS-REJECT - sending reject for id 79 to 10.10.10.10
 (client1.austin.ibm.com)
Aug 18 10:04:18 server1 radiusd[389278]: [2]:Inside proxy_response_needed() function
Aug 18 10:04:18 server1 radiusd[389278]: [2]:Proxy is not turned on
Aug 18 10:04:18 server1 radiusd[389278]: [2]:Inside get_client_secret routine for ip:10.10.10.10
Aug 18 10:04:18 server1 radiusd[389278]: [2]:Found NAS-IP = [10.10.10.10]
Aug 18 10:04:18 server1 radiusd[389278]: [2]:Found shared secret.
Aug 18 10:04:18 server1 radiusd[389278]: [2]:send_reject() Outgoing Packet:
Aug 18 10:04:18 server1 radiusd[389278]: [2]: Code = 3, Id = 79, Length = 30
Aug 18 10:04:18 server1 radiusd[389278]: [2]:send_reject() Outgoing Packet:
```

```
Aug 18 10:04:18 server1 radiusd[389278]: [2]:  Code = 3, Id = 79, Length = 30
Aug 18 10:04:18 server1 radiusd[389278]: [2]:  Authenticator = 0x05D4865C6EBEFC1A9300D2DC66F3DBE9
Aug 18 10:04:18 server1 radiusd[389278]: [2]:    Type =  18, Length =  10, Value = 0x4261646E65737321
Aug 18 10:04:18 server1 radiusd[389278]: [2]:*** Leave Process_Packet() **
```

LDAP logging

For information on LDAP logging, see the IBM Tivoli Directory Server Version 5.2 Administration Guide that can be found at Tivoli Software Information Center.

On-demand feature

You can start multiple RADIUS authentication and accounting server daemons as needed.

Each server listens on a separate port. The **radiusd.conf** file is shipped with a default port number of 1812 for authentication and 1813 for accounting. These are IANA assigned port numbers. By updating **radiusd.conf**, these port numbers, along with other ports (multiples) as needed, can be used. Be sure to use port numbers that are not assigned to existing services. When multiple port numbers are entered in the **Authentication_Ports** and **Accounting_Ports** fields in the **radiusd.conf** file, a **radiusd** daemon is started for each port. The daemons will listen on their respective port number.

Stopping and restarting RADIUS

You must stop and restart the **radiusd** daemons whenever changes are made to the RADIUS server's **/etc/radius/radiusd.conf** configuration file, or to the default authorization files, **/etc/radius/authorization/default.policy** or **/etc/radius/authorization/default.auth**. This can be handled from SMIT or from a command line.

To stop and restart the RADIUS server, use the following commands:

```
>stopsrc -s radiusd
>startsrc -s radiusd
```

Stopping and starting RADIUS is necessary because the daemon must build a memory table of all default attributes contained in the above configuration files. Shared memory is used for each local user and the local user table only gets built at daemon initialization time for performance reasons.

NLS enablement

The RADIUS **raddbm** command and the SMIT panels are NLS enabled and each uses the standard AIX NLS API calls to provide this function.

Related information

Commands: **installp**, **mkuser**, and **raddbm**

AIX Intrusion prevention

AIX intrusion prevention detects inappropriate, unauthorized or other data that might be considered harmful to a system.

The following section describes the various types of intrusion detection provided by AIX.

Related information

Commands: **chfilt**, **ckfilt**, **expfilt**, **genfilt**, **impfilt**, **lsfilt**, **mkfilt**, **mvfilt**, **rmfilt**.

Intrusion detection

This section describes the concept of intrusion detection.

Intrusion detection is the action of monitoring and analyzing system events in order to intercept and reject any attempt of unauthorized system access. In AIX, this detection of unauthorized access or attempted unauthorized access is done by observing certain actions, and then applying filter rules to these actions.

Note: You must install the **bos.net.ipsec** filesets on the host system to enable intrusion detection. The detection technologies are built upon the existing AIX Internet Protocol Security (IPsec) features.

Pattern matching filter rules

Pattern matching is the use of an IPsec filter rule for filtering networking packets. A filter pattern can be a text string, a hexadecimal string, or a file containing more than one pattern. After a pattern filter rule is established and that pattern is detected in the body of any network packet, then the predefined action of the filter rule will result.

Pattern matching filter rules only apply to inbound network packets. Use the **genfilt** command to add a filter rule to the filter rule table. The filter rules generated by this command are called manual filter rules. Use the **mkfilt** command to activate or deactivate the filter rules. The **mkfilt** command can also be used to control the filter logging function.

A pattern file can contain a list, one per line, of text patterns or hexadecimal patterns. Pattern matching filter rules can be used to guard against viruses, buffer overflows, and other network security attacks.

Pattern matching filter rules can have a negative impact on system performance if they are used too broadly, and with a high number of patterns. It is best to keep the scope of their application as narrow as possible. For example, if a known virus pattern applies to **sendmail**, then specify the **sendmail** SMTP destination port 25 in the filter rule. This allows all other traffic to pass without incurring a performance impact from pattern matching.

The **genfilt** command recognizes and understands the pattern format used in some versions found at http://www.clamav.net.

Types of patterns:

There are three basic types of patterns: text, hexadecimal, and file. Pattern matching filter rules apply to incoming packets only.

Text pattern

A text filter pattern is an ASCII string that looks similar to the following:

```
GET /../../../../../../../../
```

Hexadecimal pattern

A hexadecimal pattern looks similar to the following:

```
0x33c0b805e0cd16b807e0cd1650558becc7460200f05d0733ffb8c800b9ffff3abb00150
e670e47132c0e67158fec03c8075f033c033c9b002fa99cd26fb4183f90575f5c3
```

Note: A hexadecimal pattern is differentiated from a text pattern by the leading 0x.

Files that contain text patterns

A file can contain a list, one per line, of text patterns or hexadecimal patterns. Sample pattern files can be found at http://www.clamav.net.

Shun port and shun host filter rules

By setting a shun filter rule, you can affect a remote host or the remote host and port pair from accessing the local machine.

A shun filter rule dynamically creates an effect rule that denies the remote host or the remote host and port pair from accessing the local machine when the rule's specified criteria are met.

Because it is common for an attack to be preceded by a port scan, shun port filter rules are especially useful in preventing an intrusion by detecting this attack behavior.

For example, if the local host does not use the server port 37, which is the time server, then the remote host should not be accessing port 37, unless it is running a port scan. Place a shun port filter rule on port 37 so that if the remote host attempts to access that port, the shun filter rule creates an effective rule that blocks that host from further access for the amount of time specified shun rule **expiration time** field.

If a shun rule's **expiration time** field is set to 0, then the dynamically created effective shun rule does not expire.

Note:

1. An expiration time specified by the shun port filter rule applies only to the dynamically created effect rule.
2. Dynamically created effect rules can only be viewed with the **lsfilt -a** command.

Shun host filter rules

When the criteria of a shun host filter rule is met, then the dynamically created effective rule will block or shun all network traffic from the remote host for the specified expiration time.

Shun port filter rules

When the criteria of a shun port filter rule is met, then the dynamically created effective rule will only block or shun network traffic from this remote host's particular port, until the expiration time is exceeded.

Stateful filter rules

Stateful filters examine information such as source and destination addresses, port numbers, and status. Then, by applying IF, ELSE or ENDIF filter rules to these header flags, stateful systems can make filtering decisions in the context of an entire session rather than that of an individual packet and its header information.

Stateful inspection examines incoming and outgoing communication packets. When stateful filter rules are activated with the **mkfilt -u** command, the rules in the ELSE block are always examined until the IF rule is satisfied. After the IF rule or condition is satisfied, the rules in the IF block are used until the filter rules are reactivated with the **mkfilt -u** command.

The **ckfilt** command will check the syntax of the stateful filter rules and display them in a display in a illustrative manner such as the following:

```
%ckfilt -v4
Beginning of IPv4 filter rules.
Rule 2
IF Rule 3
    IF Rule 4
        Rule 5
    ELSE Rule 6
        Rule 7
    ENDIF Rule 8
ELSE Rule 9
    Rule 10
ENDIF Rule 11
Rule 0
```

Timed rules

Timed rules specify the amount of time, in seconds, that the filter rule is applied after it is made effective with the **mkfilt -v [4|6] -u** command.

The expiration time is specified with the **genfilt -e** command. For more information, see the **mkfilt** and **genfilt** commands.

Note: Timers have no effect on IF, ELSE or ENDIF rules. If an expiration time is specified in a shun host or shun port rule, the time applies only to the effect rule that is initiated by the shun rule. Shun rules have no expiration time.

Accessing filter rules from SMIT

This topic describes how you can configure rules from SMIT.

To configure filter rules from SMIT, complete the following steps.

1. From a command line, enter the following command:`smitty ipsec4`
2. Select **Advanced IP Security Configuration**.
3. Select **Configure IP Security Filter Rules**.
4. Select **Add an IP Security Filter Rule**.

```
                         Add an IP Security Filter Rule

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[TOP]                                                   [Entry Fields]
* Rule Action                                        [permit]            +
* IP Source Address                                  []
* IP Source Mask                                     []
  IP Destination Address                             []
  IP Destination Mask                                []
* Apply to Source Routing? (PERMIT/inbound only)     [yes]               +
* Protocol                                           [all]               +
* Source Port / ICMP Type Operation                  [any]               +
* Source Port Number / ICMP Type                     [0]                  #
* Destination Port / ICMP Code Operation             [any]               +
* Destination Port Number / ICMP Type                [0]                  #
* Routing                                            [both]              +
* Direction                                          [both]              +
* Log Control                                        [no]                +
* Fragmentation Control                              [0]                 +
* Interface                                          []                  +
  Expiration Time (sec)                              []                   #
  Pattern Type                                       [none]              +
  Pattern / Pattern File                             []
  Description                                        []


Where "Pattern Type" may be one of the following
  x   none                                                               x#
  x   pattern                                                            x
  x   file                                                               x
  x   Anti-Virus patterns
```

The choices for the action field are: `permit`, `deny`, `shun_host`, `shun_port`, `if`, `else`, `endif`.

If a pattern file is specified, then it must be readable when the filter rules are activated with the **mkfilt -a** command. The filter rules are stored in the /etc/security/ipsec_filter database.

Security checklist

This section provides a checklist of security actions to perform on a newly installed or existing system.

Although this list is not a complete security checklist, it can be used as a foundation to build a security checklist for your environment.

- When installing a new system, install AIX from secure base media. Perform the following procedures at installation time:
  - Do not install desktop software, such as CDE, GNOME, or KDE, on servers.
  - Install required security fixes and any recommended maintenance level fixes. See the eServer pSeries Support Fixes website (http://techsupport.services.ibm.com/server/fixes?view=pSeries) for the newest service bulletins, security advisories, and fix information.
  - Back up the system after the initial installation and store the system backup in a secure location.
- Establish access control lists for restricted files and directories.
- Disable unnecessary user accounts and system accounts, such as daemon, bin, sys, adm, lp, and uucp. Deleting accounts is not recommended because it deletes account information, such as user IDs and user names, which may still be associated with data on system backups. If a user is created with a previously deleted user ID and the system backup is restored on the system, the new user might have unexpected access to the restored system.
- Review the **/etc/inetd.conf**, **/etc/inittab**, **/etc/rc.nfs**, and **/etc/rc.tcpip** files on a regular basis and remove all unnecessary daemons and services.
- Verify that the permissions for the following files are set correctly:

```
-rw-rw-r-- root     system  /etc/filesystems
-rw-rw-r-- root     system  /etc/hosts
-rw------- root     system  /etc/inittab
-rw-r--r-- root     system  /etc/vfs
-rw-r--r-- root     system  /etc/security/failedlogin
-rw-rw---- root     audit   /etc/security/audit/hosts
```

- Disable the root account from being able to remotely log in. The root account should be able to log in only from the system console.
- Enable system auditing. For more information, see "Auditing overview" on page 59.
- Enable a login control policy. For more information, see "Login control" on page 17.
- Disable user permissions to run the **xhost** command. For more information, see "Managing X11 and CDE concerns" on page 23.
- Prevent unauthorized changes to the **PATH** environment variable. For more information, see "PATH environment variable" on page 32.
- Disable telnet, rlogin, and rsh. For more information, see "TCP/IP security" on page 127.
- Establish user account controls. For more information, see "User account control" on page 31.
- Enforce a strict password policy. For more information, see "Passwords" on page 38.
- Establish disk quotas for user accounts. For more information, see "Recovering from over-quota conditions" on page 45.
- Allow only administrative accounts to use the **su** command. Monitor the **su** command's logs in the **/var/adm/sulog** file.
- Enable screen locking when using X-Windows.
- Restrict access to the **cron** and **at** commands to only the accounts that need access to them.
- Use an alias for the **ls** command to show hidden files and characters in a file name.
- Use an alias for the **rm** command to avoid accidentally deleting files from the system.
- Disable unnecessary network services. For more information, see "Network services" on page 135.
- Perform frequent system backups and verify the integrity of backups.

- Subscribe to security-related e-mail distribution lists.

Security resources

This section provides information on various security-related resources such as Web sites, mailing lists and online references.

Security Web sites

AIX Virtual Private Networks: http://www-1.ibm.com/servers/aix/products/ibmsw/security/vpn/index.html)

CERIAS (Center for Education and Research in Information Assurance and Security): http://www.cerias.purdue.edu/

CERT (Computer Emergency Response Team, at Carnegie Mellon University): http://www.cert.org/

CIAC (Computer Incident Advisory Capability): http://ciac.llnl.gov/ciac/index.html

Computer Security Resource Clearinghouse: http://csrc.ncsl.nist.gov/

FIRST (Forum of Incident Response and Security Teams): http://www.first.org/

IBM eServer Security Planner: http://publib.boulder.ibm.com/infocenter/eserver/v1r1/en_US/index.htm?info/secplanr/securwiz.htm

IBM Security Solutions: http://www-3.ibm.com/security/index.shtml

OpenSSH: http://www.openssh.org/

pSeries Security home page: http://www-1.ibm.com/servers/eserver/pseries/security/

Security mailing lists

CERT: http://www.cert.org/contact_cert/

IBM eServer pSeries Support Subscription Service: https://techsupport.services.ibm.com/server/pseries.subscriptionSvcs

comp.security.unix: news:comp.security.unix

Security online references

Common Criteria Concepts FAQ: http://www.radium.ncsc.mil/tpep/process/faq-sect3.html

Rainbow Series Library: http://www.radium.ncsc.mil/tpep/library/rainbow/

faqs.org: http://www.faqs.org/faqs/computer-security/

IBM AIX Information Center: http://publib16.boulder.ibm.com/pseries/index.htm

Summary of common AIX system services

The following table lists the more common system services within AIX. Use this table to recognize a starting point for securing your system.

Before you secure your system, back up all your original configuration files, especially the following:

- **/etc/inetd.conf**
- **/etc/inittab**
- **/etc/rc.nfs**
- **/etc/rc.tcpip**

Service	Daemon	Started by	Function	Comments
inetd/bootps	inetd	/etc/inetd.conf	bootp services to diskless clients	• Necessary for Network Installation Management (NIM) and remote booting of systems • Works concurrently with tftp • Disable in most cases
inetd/chargen	inetd	/etc/inetd.conf	character generator (testing only)	• Available as a TCP and UDP service • Provides opportunity for Denial of Service attacks • Disable unless you are testing your network
inetd/cmsd	inetd	/etc/inetd.conf	calendar service (as used by CDE)	• Runs as root, therefore a security concern • Disable unless you require this service with CDE • Disable on back room database servers
inetd/comsat	inetd	/etc/inetd.conf	Notifies incoming electronic mail	• Runs as root, therefore a security concern • Seldom required • Disable
inetd/daytime	inetd	/etc/inetd.conf	obsolete time service (testing only)	• Runs as root • Available as a TCP and UDP service • Provides opportunity for a Denial of Service PING attacks • Service is obsolete and used for testing only • Disable
inetd/discard	inetd	/etc/inetd.conf	/dev/null service (testing only)	• Available as TCP and UDP service • Used in Denial of Service Attacks • Service is obsolete and used for testing only • Disable

Service	Daemon	Started by	Function	Comments
inetd/dtspc	inetd	/etc/inetd.conf	CDE Subprocess Control	• This service is started automatically by the **inetd** daemon in response to a CDE client requesting a process to be started on the daemon's host. This makes it vulnerable to attacks • Disable on back room servers with no CDE • CDE might be able to function without this service • Disable unless absolutely needed
inetd/echo	inetd	etc/inetd.conf	echo service (testing only)	• Available as UDP and TCP service • Could be used in Denial of Service or Smurf attacks • Used to echo at someone else to get through a firewall or start a datastorm • Disable
inetd/exec	inetd	/etc/inetd.conf	remote execution service	• Runs as root user • Requires that you enter a user ID and password, which are passed unprotected • This service is highly susceptible to being snooped • Disable
inetd/finger	inetd	/etc/inetd.conf	finger peeking at users	• Runs as root user • Gives out information about your systems and users • Disable
inetd/ftp	inetd	/etc/inetd.conf	file transfer protocol	• Runs as root user • User id and password are transferred unprotected, thus allowing them to be snooped • Disable this service and use a public domain secure shell suite
inetd/imap2	inetd	/etc/inetd.conf	Internet Mail Access Protocol	• Ensure that you are using the latest version of this server • Only necessary if you are running a mail server. Otherwise, disable • User ID and password are passed unprotected
inetd/klogin	inetd	/etc/inetd.conf	Kerberos login	• Enabled if your site uses Kerberos authentication
inetd/kshell	inetd	/etc/inetd.conf	Kerberos shell	• Enabled if your site uses Kerberos authentication

Service	Daemon	Started by	Function	Comments
inetd/login	inetd	/etc/inetd.conf	rlogin service	• Susceptible to IP spoofing, DNS spoofing • Data, including User IDs and passwords, is passed unprotected • Runs as root user • Use a secure shell instead of this service
inetd/netstat	inetd	/etc/inetd.conf	reporting of current network status	• Could potentially give network information to hackers if run on your system • Disable
inetd/ntalk	inetd	/etc/inetd.conf	Allows users to talk with each other	• Runs as root user • Not required on production or back room servers • Disable unless absolutely needed
inetd/pcnfsd	inetd	/etc/inetd.conf	PC NFS file services	• Disable service if not currently in use • If you need a service similar to this, consider Samba, as the pcnfsd daemon predates Microsoft's release of SMB specifications
inetd/pop3	inetd	/etc/linetd.conf	Post Office Protocol	• User IDs and passwords are sent unprotected • Only needed if your system is a mail server and you have clients who are using applications that only support POP3 • If your clients use IMAP, use that instead, or use the POP3s service. This service has a Secure Socket Layer (SSL) tunnel • Disable if you are not running a mail server or have clients who need POP services
inetd/rexd	inetd	/etc/inetd.conf	remote execution	• Runs as root user • Peers with the **on** command • Disable service • Use **rsh**and **rshd** instead

Service	Daemon	Started by	Function	Comments
inetd/quotad	inetd	/etc/inetd.conf	reports of file quotas (for NFS clients)	• Only needed if you are running NFS file services • Disable this service unless required to provide an answer for the **quota** command • If you need to use this service, keep all patches and fixes for this service up to date
inetd/rstatd	inetd	/etc/inetd.conf	Kernel Statistics Server	• If you need to monitor systems, use SNMP and disable this service • Required for use of the **rup** command
inetd/rusersd	inetd	/etc/inetd.conf	info about user logged in	• This is not an essential service. Disable • Runs as root user • Gives out a list of current users on your system and peers with rusers
inetd/rwalld	inetd	/etc/inetd.conf	write to all users	• Runs as root user • If your systems have interactive users, you might need to keep this service • If your systems are production or database servers, this is not needed • Disable
inetd/shell	inetd	/etc/inetd.conf	rsh service	• Disable this service if possible. Use Secure Shell instead • If you must use this service, use the TCP Wrapper to stop spoofing and limit exposures • Required for the**Xhier** software ditribution program
inetd/sprayd	inetd	/etc/inetd.conf	RPC spray tests	• Runs as root user • Might be required for diagnosis of NFS network problems • Disable if you are not running NFS
inetd/systat	inetd	/etc/inted.conf	″ps -ef″ status report	• Allows for remote sites to see the process status on your system • This service is disabled by default. You must check periodically to ensure that the service has not been enabled

Service	Daemon	Started by	Function	Comments
inetd/talk	inetd	/etc/inetd.conf	establish split screen between 2 users on the net	• Not a required service • Used with the **talk** command • Provides UDP service at Port 517 • Disable unless you need multiple interactive chat sessions for UNIX user
inetd/ntalk	inetd	/etc/inetd.conf	″new talk″ establish split screen between 2 users on the net	• Not a required service • Used with the **talk** command • Provides UDP service at Port 517 • Disable unless you need multiple interactive chat sessions for UNIX user
inetd/telnet	inetd	/etc/inetd.conf	telnet service	• Supports remote login sessions, but the password and ID are passed unprotected • If possible, disable this service and use Secure Shell for remote access instead
inetd/tftp	inetd	/etc/inetd.conf	trivial file transfer	• Provides UDP service at port 69 • Runs as root user and might be compromised • Used by NIM • Disable unless you are using NIM or have to boot a diskless workstation
inetd/time	inetd	/etc/inetd.conf	obsolete time service	• Internal function of **inetd** that is used by **rdate** command. • Available as TCP and UDP service • Sometimes used to synchronize clocks at boot time • Service is outdated. Use ntpdate instead • Disable this only after you have tested your systems (boot/reboot) with this service disabled and have observed no problems
inetd/ttdbserver	inetd	/etc/inetd.conf	tool-talk database server (for CDE)	• The **rpc.ttdbserverd** runs as root user and might be compromised • Stated as a required service for CDE, but CDE is able to work without it • Should not be run on back room servers or any systems where security is a concern

Service	Daemon	Started by	Function	Comments
inetd/uucp	inetd	/etc/inetd.conf	UUCP network	• Disable unless you have an application that uses UUCP
inittab/dt	init	/etc/rc.dt script in the /etc/inittab	desktop login to CDE environment	• Starts the X11 server on the console • Supports the X11 Display Manager Control Protocol (xdcmp) so that other X11 stations can log into the same machine • Service should be used on personal workstations only. Avoid using it for back room systems
inittab/dt_nogb	init	/etc/inittab	desktop login to CDE environment (NO graphic boot)	• No graphical display until the system is up fully • Same concerns as inittab/dt
inittab/httpdlite	init	/etc/inittab	web server for the **docsearch** command	• Default web server for the docsearch engine • Disable unless your machine is a documentation server
inittab/i4ls	init	/etc/inittab	license manager servers	• Enable for development machines • Disable for production machines • Enable for back room database machines that have license requirements • Provides support for compilers, database software, or any other licensed products
inittab/imqss	init	/etc/inittab	search engine for ″docsearch″	• Part of the default web server for the docsearch engine • Disable unless your machine is a documentation server
inittab/lpd	init	/etc/inittab	BSD line printer interface	• Accepts print jobs from other systems • You can disable this service and still send jobs to the print server • Disable this after you confirm that printing is not affected
inittab/nfs	init	/etc/inittab	Network File System/Net Information Services	• NFS and NIS services based which were built on UDP/RPC • Authentication is minimal • Disable this for back room machines

Service	Daemon	Started by	Function	Comments
inittab/piobe	init	/etc/inittab	printer IO Back End (for printing)	• Handles the scheduling, spooling and printing of jobs submitted by the **qdaemon** daemon • Disable if you are not printing from your system because you are sending print job to a server
inittab/qdaemon	init	/etc/inittab	queue daemon (for printing	• Submits print jobs to the **piobe** daemon • If you are not printing from your system, then disable
inittab/uprintfd	init	/etc/inittab	kernel messages	• Generally not required • Disable
inittab/writesrv	init	/etc/inittab	writing notes to ttys	• Only used by interactive UNIX workstation users • Disable this service for servers, back room databases, and development machines • Enable this service for workstations
inittab/xdm	init	/etc/inittab	traditional X11 Display Management	• Do not run on back room production or database servers • Do not run on development systems unless X11 display management is needed • Acceptable to run on workstations if graphics are needed
rc.nfs/automountd		/etc/rc.nfs	automatic file systems	• If you use NFS, enable this for workstations • Do not use the automounter for development or back room servers
rc.nfs/biod		/etc/rc.nfs	Block IO Daemon (required for NFS server)	• Enabled for NFS server only • If not an NFS server, then disable this along with nfsd and rpc.mountd
rc.nfs/keyserv		/etc/rc.nfs	Secure RPC Key server	• Manages the keys required for secure RPC • Important for NIS+ • Disable this if you are *not* using NFS and NIS and NIS+
rc.nfs/nfsd		/etc/rc.nfs	NFS Services (required for NFS Server)	• Authentication is weak • Can lend itself to stack frame crashing • Enable if on NFS file servers • If you disable this, then disable **biod**, **nfsd**, and **rpc.mountd** as well

Service	Daemon	Started by	Function	Comments
rc.nfs/rpc.lockd		/etc/rc.nfs	NFS file locks	• Disable if you are not using NFS • Disable this if you are not using file locks across the network • **lockd** daemon is mentioned in the SANS Top Ten Security Threats
rc.nfs/rpc.mountd		/etc/rc.nfs	NFS file mounts (required for NFS Server)	• Authentication is weak • Can lend itself to stack frame crashing • Should be enabled only on NFS file servers • If you disable this, then disable **biod** and **nfsd** as well
rc.nfs/rpc.statd		/etc/rc.nfs	NFS file locks (to recover them)	• Implements file locks across NFS • Disable unless you are using NFS
rc.nfs/rpc.yppasswdd		/etc/rc.nfs	NIS password daemon (for NIS master)	• Used to manipulate the local password file • Only required when the machine in question is the NIS master; disable in all other cases
rc.nfs/ypupdated		/etc/rc.nfs	NIS Update daemon (for NIS slave)	• Receives NIS database maps pushed from the NIS Master • Only required when the machine in question is a NIS slave to a Master NIS Server
rc.tcpip/autoconf6		/etc/rc.tcpip	IPv6 interfaces	• Disable unless you are running IP Version 6
rc.tcpip/dhcpcd		/etc/rc.tcpip	Dynamic Host Configure Protocol (client)	• Back room servers should not rely on DHCP. Disable this service • If your host is not using DHCP, disable
rc.tcpip/dhcprd		/etc/rc.tcpip	Dynamic Host Configure Protocol (relay	• Grabs DHCP broadcasts and sends them to a server on another network • Duplicate of a service found on routers • Disable this if you are not using DHCP or rely on passing information between networks

Service	Daemon	Started by	Function	Comments
rc.tcpip/dhcpsd		/etc/rc.tcpip	Dynamic Host Configure Protocol (server	• Answers DHCP requests from clients at boot time; gives client information, such as IP name, number, netmask, router, and broadcast address • Disable this if you are not using DHCP • Disabled on production and back room servers along with hosts not using DHCP
rc.tcpip/dpid2		/etc/rc.tcpip	outdated SNMP service	• Disable unless you need SNMP
rc.tcpip/gated		/etc.rc.tcpip	gated routing between interfaces	• Emulates router function • Disable this service and use RIP or a router instead
rc.tcpip/inetd		/etc/rc.tcpip	inetd services	• A thoroughly secured system should have this disabled, but is often not practical • Disabling this will disable remote shell services which are required for some mail and web servers
rc.tcpip/mrouted		/etc/rc.tcpip	multi-cast routing	• Emulates router function of sending multi-cast packets between network segments • Disable this service. Use a router instead
rc.tcpip/names		/etc/rc.tcpip	DNS name server	• Use this only if your machine is a DNS name server • Disable for workstation, development and production machines
rc.tcpip/ndp-host		/etc/rc.tcpip	IPv6 host	• Disable unless you use IP Version 6
rc.tcpip/ndp-router		/etc/rc.tcpip	IPv6 routing	• Disable this unless you use IP Version 6. Consider using a router instead of IP Version 6
rc.tcpip/portmap		/etc/rc.tcpip	RPC services	• Required service • RPC servers register with **portmap** daemon. Clients who need to locate RPC services ask the **portmap** daemon to tell them where a particular service is located • Disable only if you have managed to reduce RPC service so that the only one remaining is **portmap**

Service	Daemon	Started by	Function	Comments
rc.tcpip/routed		/etc/rc.tcpip	RIP routing between interfaces	• Emulates router function • Disable if you have a router for packets between networks
rc.tcpip/rwhod		/etc/rc.tcpip	Remote ″who″ daemon	• Collects and broadcasts data to peer servers on the same network • Disable this service
rc.tcpip/sendmail		/etc/rc.tcpip	mail services	• Runs as root user • Disable this service unless the machine is used as a mail server • If disabled, then do one of the following: – Place an entry in crontab to clear the queue. Use the **/usr/lib/sendmail -q** command – Configure DNS services so that the mail for your server is delivered to some other system
rc.tcpip/snmpd		/etc/rc.tcpip	Simple Network Management Protocol	• Disable if you are not monitoring the system via SNMP tools • SNMP may be required on critical servers
rc.tcpip/syslogd		/etc/rc.tcpip	system log of events	• Disabling this service is *not* recommended • Prone to denial of service attacks • Required in any system
rc.tcpip/timed		/etc/rc.tcpip	Old Time Daemon	• Disable this service and use xntp instead
rc.tcpip/xntpd		/etc/rc.tcpip	New Time Daemon	• Keeps clocks on systems in sync • Disable this service. • Configure other systems as time servers and let other systems synchronize to them with a cron job that calls ntpdate
dt login		/usr/dt/config/Xaccess	unrestricted CDE	• If you are not providing CDE login to a group of X11 stations, you can restrict dtlogin to the console.

Service	Daemon	Started by	Function	Comments
anonymous FTP service		user rmuser -p <username>	anonymous ftp	• Anonymous FTP ability prevents you from tracing FTP usage to a specific user • Remove user ftp if that user account exists, as follows: **rmuser -p ftp** • Further security can be obtained by populating the **/etc/ftpusers** file with a list of those who should not be able to ftp to your system
anonymous FTP writes			anonymous ftp uploads	• No file should belong to ftp. • FTP anonymous uploads allow the potential for misbehaving code to be placed on your system. • Put the names of those users you want to disallow into the **/etc/ftpusers** file • Some examples of system-created users you might want to disallow from anonymously uploading via FTP to your system are: root, daemon, bin.sys, admin.uucp, guest, nobody, lpd, nuucp, ladp • Change the owner and group rights to the **ftpusers** files as follows: **chown root:system /etc/ftpusers** • Change the permissions to the **ftpusers** files to a stricter setting as follows: **chmod 644 /etc/ftpusers**
ftp.restrict			ftp to system accounts	• No user from the outside should be allowed to replace root files using **ftpusers** file
root.access		/etc/security/user	rlogin/telnet to root account	• Set the rlogin option in the **etc/security/user** file to false • Anyone logging in as root should first log in under their own name and then **su** to root; this provides an audit trail
snmpd.readWrite		/etc/snmpd.conf	SNMP readWrite communities	• If you are *not* using SNMP, disable the SNMP daemon. • Disable community private and community system in the **/etc/snmpd.conf** file • Restrict 'public' community to those IP addresses that are monitoring your system

Service	Daemon	Started by	Function	Comments
syslog.conf			configure syslogd	• If you have not configured **/etc/syslog.conf**, then disable this daemon • If you are using **syslog.conf** to log system messages, then keep enabled

Summary of network service options

To achieve a higher level of system security, there are several network options that you can change using `0` to disable and `1` to enable. The following list identifies these parameters you can use with the **no** command.

Parameter	Command	Purpose
bcastping	/usr/sbin/no -o bcastping=0	Allows response to ICMP echo packets to the broadcast address. Disabling this prevents Smurf attacks.
clean_partial_conns	/usr/sbin/no -o clean_partial_conns=1	Specifies whether or not SYN (synchronizes the sequence number) attacks are being avoided.
directed_broadcast	/usr/sbin/no -o directed_broadcast=0	Specifies whether to allow a directed broadcast to a gateway. Setting to 0 helps prevent directed packets from reaching a remote network.
icmpaddressmask	/usr/sbin/no -o icmpaddressmask=0	Specifies whether the system responds to an ICMP address mask request. Disabling this prevents access through source routing attacks.
ipforwarding	/usr/sbin/no -o ipforwarding=0	Specifies whether the kernel should forward packets. Disabling this prevents redirected packets from reaching remote network.
ipignoreredirects	/usr/sbin/no -o ipignoreredirects=1	Specifies whether to process redirects that are received.
ipsendredirects	/usr/sbin/no -o ipsendredirects=0	Specifies whether the kernel should send redirect signals. Disabling this prevents redirected packets from reaching remote network.
ip6srcrouteforward	/usr/sbin/no -o ip6srcrouteforward=0	Specifies whether the system forwards source-routed IPv6 packets. Disabling this prevents access through source routing attacks.
ipsrcrouteforward	/usr/sbin/no -o ipsrcrouteforward=0	Specifies whether the system forwards source-routed packets. Disabling this prevents access through source routing attacks.
ipsrcrouterecv	/usr/sbin/no -o ipsrcrouterecv=0	Specifies whether the system accepts source-routed packets. Disabling this prevents access through source routing attacks
ipsrcroutesend	/usr/sbin/no -o ipsrcroutesend=0	Specifies whether applications can send source-routed packets. Disabling this prevents access through source routing attacks.

Parameter	Command	Purpose
nonlocsroute	/usr/sbin/no -o nonlocsrcroute=0	Tells the Internet Protocol that strictly source-routed packets may be addressed to hosts outside the local network. Disabling this prevents access through source routing attacks.
tcp_icmpsecure	/usr/sbin/no -o tcp_icmpsecurer=1	Protects TCP connections against ICMP (Internet Control Message Protocol) source quench and PMTUD (Path MTU Discovery) attacks. Checks the payload of the ICMP message to test the sequence number of the TCP header is within the range of acceptable sequence numbers. Values: 0=off (default); 1=on.
ip_nfrag	/usr/sbin/no -o ip_nfrag=200	Specifies the maximum number of fragments of an IP packet that can be kept on the IP reassembly queue at a time (default value of 200 keeps up to 200 fragments of an IP packet in the IP reassembly queue).
tcp_pmtu_discover	/usr/sbin/no -o tcp_pmtu_discover=0	Disabling this prevents access through source routing attacks.
tcp_tcpsecure	/usr/sbin/no -o tcp_tcpsecure=7	Protects TCP connections against vulnerabilities. Values: 0=no protection; 1=sending a fake SYN to an established connection; 2=sending a fake RST to an established connection; 3=injecting data in an established TCP connection; 5–7=combination of the above vulnerabilities.
udp_pmtu_discover	/usr/sbin/no -o udp_pmtu_discover=0	Enables or disables path MTU discovery for TCP applications. Disabling this prevents access through source routing attacks.

For more information about network-tunable options, see *AIX 5L Version 5.3 Performance Management Guide*.

Index

Vos remarques sur ce document / Technical publication remark form

Titre / Title : Bull AIX 5L Security Guide

Nº Reférence / Reference Nº : 86 A2 57EM 02

Daté / Dated : October 2005

ERREURS DETECTEES / ERRORS IN PUBLICATION

AMELIORATIONS SUGGEREES / SUGGESTIONS FOR IMPROVEMENT TO PUBLICATION

Vos remarques et suggestions seront examinées attentivement.
Si vous désirez une réponse écrite, veuillez indiquer ci-après votre adresse postale complète.

Your comments will be promptly investigated by qualified technical personnel and action will be taken as required.
If you require a written reply, please furnish your complete mailing address below.

NOM / NAME : ______________________________ Date : __________

SOCIETE / COMPANY : ______________________________

ADRESSE / ADDRESS : ______________________________

____

Remettez cet imprimé à un responsable BULL ou envoyez-le directement à :

Please give this technical publication remark form to your BULL representative or mail to:

BULL CEDOC
357 AVENUE PATTON
B.P.20845
49008 ANGERS CEDEX 01
FRANCE

Technical Publications Ordering Form

Bon de Commande de Documents Techniques

To order additional publications, please fill up a copy of this form and send it via mail to:
Pour commander des documents techniques, remplissez une copie de ce formulaire et envoyez-la à :

BULL CEDOC
ATTN / Mr. L. CHERUBIN
357 AVENUE PATTON
B.P.20845
49008 ANGERS CEDEX 01
FRANCE

Phone / Téléphone : +33 (0) 2 41 73 63 96
FAX / Télécopie +33 (0) 2 41 73 60 19
E–Mail / Courrier Electronique : srv.Cedoc@franp.bull.fr

Or visit our web sites at: / Ou visitez nos sites web à:

http://www.logistics.bull.net/cedoc
http://www-frec.bull.com   http://www.bull.com

CEDOC Reference # Nº Référence CEDOC	**Qty** Qté	**CEDOC Reference #** Nº Référence CEDOC	**Qty** Qté	**CEDOC Reference #** Nº Référence CEDOC	**Qty** Qté
__ __ ____ _ [__]		__ __ ____ _ [__]		__ __ ____ _ [__]	
__ __ ____ _ [__]		__ __ ____ _ [__]		__ __ ____ _ [__]	
__ __ ____ _ [__]		__ __ ____ _ [__]		__ __ ____ _ [__]	
__ __ ____ _ [__]		__ __ ____ _ [__]		__ __ ____ _ [__]	
__ __ ____ _ [__]		__ __ ____ _ [__]		__ __ ____ _ [__]	
__ __ ____ _ [__]		__ __ ____ _ [__]		__ __ ____ _ [__]	
__ __ ____ _ [__]		__ __ ____ _ [__]		__ __ ____ _ [__]	
[__] : **no revision number means latest revision** / pas de numéro de révision signifie révision la plus récente					

NOM / NAME : ________________________________ Date : __________

SOCIETE / COMPANY : ________________________________

ADRESSE / ADDRESS : ________________________________

____

PHONE / TELEPHONE : ____________________ FAX : ____________________

E–MAIL : ________________________________

For Bull Subsidiaries / Pour les Filiales Bull :

Identification: ________________________________

For Bull Affiliated Customers / Pour les Clients Affiliés Bull :

Customer Code / Code Client : ________________________________

For Bull Internal Customers / Pour les Clients Internes Bull :

Budgetary Section / Section Budgétaire : ________________________________

For Others / Pour les Autres :

Please ask your Bull representative. / Merci de demander à votre contact Bull.

BULL CEDOC
357 AVENUE PATTON
B.P.20845
49008 ANGERS CEDEX 01
FRANCE

ORDER REFERENCE
86 A2 57EM 02