Installing and Administering the CIFS/9000 Client

Version A.01.08

Manufacturing Part Number : B8724-90022
AR0902

U.S.A.

Legal Notices

The information in this document is subject to change without notice.

Hewlett-Packard makes no warranty of any kind with regard to this manual, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Hewlett-Packard shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Warranty. A copy of the specific warranty terms applicable to your Hewlett- Packard product and replacement parts can be obtained from your local Sales and Service Office.

Restricted Rights Legend. Use, duplication or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 for DOD agencies, and subparagraphs (c) (1) and (c) (2) of the Commercial Computer Software Restricted Rights clause at FAR 52.227-19 for other agencies.

HEWLETT-PACKARD COMPANY
3000 Hanover Street
Palo Alto, California 94304 U.S.A.

Use of this manual and flexible disk(s) or tape cartridge(s) supplied for this pack is restricted to this product only. Additional copies of the programs may be made for security and back-up purposes only. Resale of the programs in their present form or with alterations, is expressly prohibited.

PAM NTLM includes a library derived from the Open Source Samba product. This library is subject to the GPL license. For detailed information, refer to the GPL license in Chapter 5 of the CIFS/9000 Server manual.

Trademark Notices. UNIX is a registered trademark of The Open Group.

Contents

Contents

Preface

The information in this manual is intended for network managers or network security administrators who install and administer the CIFS/9000 Client.

This manual describes how to install, configure, and troubleshoot the HP CIFS/9000 Client software product on HP 9000 systems.

The manual is organized as follows:

Chapter 1	“Introduction to the CIFS/9000 Client” describes the CIFS/9000 Client product history, features, requirements and limitations.
Chapter 2	“Installing and Configuring the CIFS/9000 Client” describes how to install, configure and verify the CIFS/9000 client software.
Chapter 3	“Commandline Utilities” provides Unix man-pages for all CIFS/9000 Client utilities.
Chapter 4	“Troubleshooting the CIFS/9000 Client” describes detailed procedures to help diagnose CIFS/9000 Client problems.
Chapter 5	“CIFS/9000 Configuration File” provides a list of all configuration variables if you want to customize CIFS/9000 software.
Chapter 6	“CIFS Authentication (PAM module)” provides detailed information about the NTLM PAM authentication service.

About This Document

The technical contents of this document are identical to that of the IA1122 (June 2002) release.

As of this release, the A.01.08 version of the CIFS Client supports the following versions of HP-UX:

- HP-UX 11.0
- HP-UX 11i
- HP-UX 11.22

1 Introduction to the CIFS/9000 Client

This chapter provides a CIFS/9000 Client description.

It contains the following sections:

- Introduction to CIFS/9000.
- CIFS/9000 Client Description.
- CIFS/9000 Client Features.

Introduction to CIFS/9000

CIFS/9000 provides HP-UX with a distributed file system based on the Microsoft Common Internet File System (CIFS) protocols. CIFS/9000 implements both the server and client components of the CIFS protocol on HP-UX.

The CIFS/9000 Server is based on the well-established open-source software Samba, version 2.0.6, and provides file and print services to CIFS clients including Windows 95, 98, NT, 2000, and HP-UX machines running CIFS/9000 Client software.

The CIFS/9000 Client enables HP-UX users to mount as UNIX filesystems shares from CIFS file servers including Windows servers and HP-UX machines running CIFS/9000 Server. The CIFS/9000 client also offers an optional Pluggable Authentication Module (PAM) that implements the Windows NTLM authentication protocols. When installed and configured within HP-UX's PAM facility, PAM NTLM allows HP-UX users to be authenticated against a Windows authentication server.

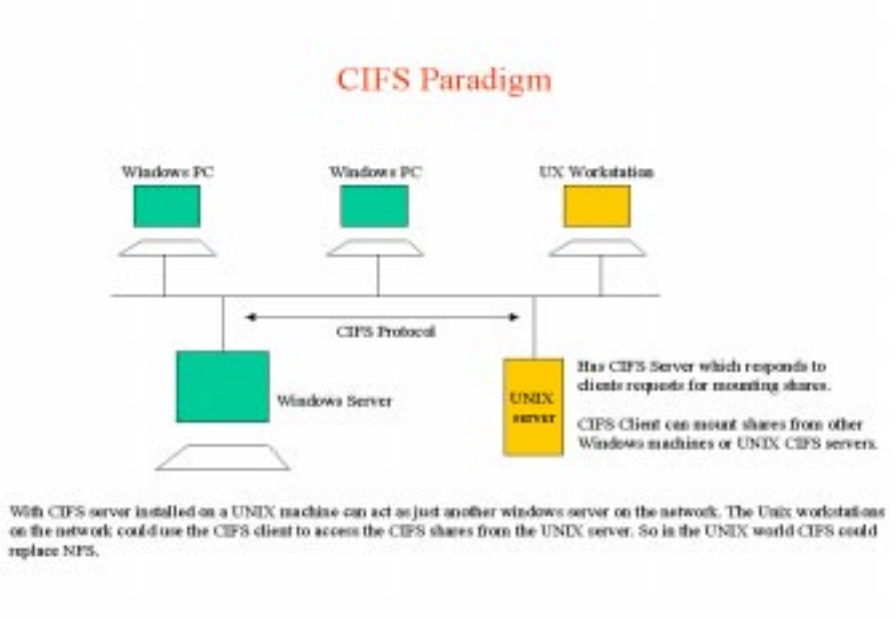
What is the CIFS Protocol?

CIFS had its beginnings in the networking protocols, sometimes called Server Message Block (SMB) protocols, that were developed in the late 1980's for PCs to share files over the then nascent Local Area Network technologies (e.g., Ethernet). SMB is the native file-sharing protocol in the Microsoft Windows 95, Windows NT, and OS/2 operating systems and the standard way that millions of PC users share files across corporate intranets.

CIFS is simply a renaming of SMB; and CIFS and SMB are, for all practical purposes, one and the same. (Microsoft now emphasizes the use of "CIFS," although references to "SMB" still occur.) CIFS is also widely available on UNIX, VMS(tm), Macintosh, and other platforms.

Despite its name, CIFS is not actually a file system unto itself. More accurately, CIFS is a remote file access protocol; it provides access to files on remote systems. It sits on top of and works with the file systems of its host systems. CIFS defines both a server and a client: the CIFS client is used to access files on a CIFS server.

CIFS/9000 speaks the CIFS protocol from the HP-UX machines, which enables directories from HP-UX servers to be mounted on to Windows machines and vice versa.



PAM NTLM

The HP-UX PAM subsystem gives system administrators the flexibility of choosing any authentication service available on the system to perform authentication. The framework also allows new authentication service modules to be plugged in and made available without modifying the applications.

The PAM framework, *libpam*, consists of an interface library and multiple authentication service modules. The authentication service modules are a set of dynamically loadable objects invoked by the PAM API to provide a particular type of user authentication.

NT LAN Manager (NTLM) is the protocol by which CIFS clients are authenticated by CIFS servers. PAM NTLM is a PAM module that implements the NTLM protocol. It enables users logging in to an HP-UX system to have access to CIFS-mounted file systems without having to use the cifslogin command.

CIFS/9000 Client Description

CIFS/9000 Client implements the CIFS protocols on HP-UX such that HP-UX users may mount shares from CIFS servers as UNIX file systems.

CIFS/9000 Client Features

Following is a list of the CIFS/9000 Client major features:

- CIFS UNIX Extensions
- NTLM PAM Integration
- Client -side Caching.
- Support for Internationalized Clients

CIFS UNIX Extensions

CIFS UNIX Extensions enable the CIFS Client and Samba server to implement standard UNIX file system features. These include:

- UNIX permission modes
- File ownership based on UNIX UIDs, and GIDs
- Symbolic links and hard links
- Standard UNIX timestamps for file access, change, and modification.
- Includes other data contained in the UNIX stat(2) data structure.

NOTE This feature only works with CIFS servers that support CIFS UNIX extensions. HP Samba is the only known CIFS server that supports CIFS extensions at this time.

NTLM PAM Integration

NT LAN Manager (NTLM) is the protocol by which CIFS clients are authenticated by CIFS servers. When used in conjunction with HP's NTLM Pluggable Authentication Module (PAM) and the CIFS/9000 Client, users who log in to an HP-UX system will have access automatically to CIFS-mounted file systems provided that PAM NTLM and the CIFS server are using the same database.

Client- Side Caching

CIFS/9000 client supports Opportunistic Locks as described in the CIFS specification to provide client side data caches and read ahead. This may improve the network performance by an order of magnitude.

Support for Internationalized Clients

The CIFS Client is designed to work with a variety of internationalized clients and servers. It can use Unicode to transmit multi-byte characters on the network, or any of several character encoding tables located in */etc/opt/cifsclient/unitables*. See the *README* file in that directory for an index of the tables.

2 Installing and Configuring the CIFS/9000 Client

This chapter describes the procedures to install HP CIFS/9000 Client software onto your system.

It contains the following sections:

- CIFS/9000 Client Requirements and Limitations.
- Overview of HP CIFS/9000 Client Installation and Configuration.
- Step 1: Checking HP CIFS/9000 Client Installation Prerequisites.
- Step 2: Installing HP CIFS/9000 Client and PAM Software.
- Step 3: Configuring the CIFS/9000 Client Configuration.
- Step 4: Starting and Stopping the CIFS/9000 Client Daemon.

CIFS/9000 Client Requirements and Limitations

System Requirements

The CIFS/9000 Client runs on all HP Workstations and Servers that are capable of running HP-UX version 11.0 or later, in either 32 or 64-bit mode. No specific system patches are required for the CIFS/9000 Client.

Overview of HP CIFS/9000 Client Installation and Configuration

Installation of the HP CIFS/9000 client includes checking installation prerequisites, loading the HP CIFS/9000 Client filesets using the *swinstall(1M)* utility, and completing CIFS/9000 configuration procedures.

The CIFS Client software and PAM NTLM software are delivered as two separate products, packaged for installation via HP Software Distributor (SD). HP recommends that both products be installed at the same time. This is not a requirement as each one can also be installed and run as a standalone product. To install and remove software, use the HP-UX commands swinstall(1M) and swremove(1M). Detailed information on these commands are provided in the HP-UX man pages.

The CIFS Client forces a system reboot during installation and removal. The CIFS Client modifies the kernel so that it will recognize CIFS as a mountable filesystem.

When you install the bundles for the CIFS/9000 Client, there will be two products for you to install. The first one is the CIFS/9000 client software and the second one (optional) is the NTLM PAM module.

NOTE You can download the CIFS/9000 Client software from www.software.hp.com.

Step 1: Checking HP CIFS/9000 Client Installation Prerequisites

Prior to loading the CIFS/9000 Client software onto your system, check that you have met the following hardware and software prerequisites.

1. Check that you have the PAM library patches.

   To obtain information about the OS, execute the command:

   ```
   uname -a
   ```

   To obtain information about a patch, execute the command:

   ```
   swlist -i
   ```

   Refer to the CIFS/9000 Client release note for information about patch dependence.

2. You must be “root” to perform the installation.

Step 2: Installing HP CIFS/9000 Client and PAM Software

Follow the steps below to load HP CIFS/9000 Client software using the HP-UX *swinstall* program.

1. Log in as `root`.
2. Insert the software media (disk) into the appropriate drive.
3. Run the *swinstall* program using the command:

   `swinstall`

   This opens the Software Selection Window and Specify Source Window.
4. Change the Source Host Name if necessary, enter the mount point of the drive in the Source Depot Path field, and activate the **OK** button to return to the Software Selection Window. Activate the Help button to get more information.

   The Software Selection Window now contains a list of available software bundles to install.
5. Highlight the HP CIFS/9000 Client software and press Enter.
6. Highlight one or both products and press Enter.

   There will be two software products for the client. One is for the Client software and the other is for the NTLM PAM plug-in authentication module.
7. Choose `Mark for Install` from the "Actions" menu to choose the product to be installed.
8. Choose `Install` from the "Actions" menu to begin product installation and open the Install Analysis Window.
9. Activate the **OK** button in the Install Analysis Window when the Status field displays a Ready message.
10. Activate the **Yes** button at the Confirmation Window to confirm that you want to install the software. *swinstall* displays the Install Window.

View the Install Window to read processing data while the software is being installed. When the Status field indicates Ready and the Note Window opens.

swinstall loads the fileset, runs the control scripts for the fileset, and builds the kernel. Estimated time for processing: 3 to 5 minutes.

11. Activate the **OK** button on the Note Window to reboot the system.

    The user interface disappears and the system reboots.

12. When the system reboots, check the log files in */var/adm/sw/swinstall.log* and */var/adm/sw/swagent.log* to make sure the installation was successful.

Step 3: Configuring the CIFS/9000 Client

The configuration file for the CIFS/9000 Client, */etc/opt/cifsclient/cifsclient.cfg*, can be used as delivered, with no modification of its default values.

Editing *cifsclient.cfg*.

If appropriate, edit the CIFS/9000 client configuration file */etc/opt/cifsclient/cifsclient.cfg as described below.*

1. Update the domain variable with the name of the NT domain to which the client will belong. This step is recommended, but not required.

   domain = hpnet_dom

2. Configure Internationalized Clients.

   The CIFS Client is designed to work with a variety of internationalized clients and servers. It can use Unicode to transmit multi-byte characters on the network, or any of several character encoding tables located in */etc/opt/cifsclient/unitables*. See the *README* file in that directory for an index of the tables.

   Each table is a "CharMap" file which can be configured for encoding file and directory names on the client or server (file contents are left untouched). The character-set displayed on the CIFS Client console is configured with the parameter *clientCharMapFile*, which selects any one of the many character mapping files provided with the product. Character translations for communications with CIFS Servers can be done either in Unicode or through the configuration parameter *serverCharMapFile*, which also is used to select a character mapping file. Use of Unicode is turned on and off with the *useUnicode* parameter.

   The default settings in *cifsclient.cfg*, are:

   ```
   serverCharMapFile = "/etc/opt/cifsclient/unitables/unimapCP437.cfg";
   clientCharMapFile = "/etc/opt/cifsclient/unitables/unimap8859-1.cfg";
   ```

   If, for example, your CIFS Client is configured as a Japanese system using the *Shift-JIS* locale, and it is connected to a Japanese CIFS Server that also uses *Shift-JIS*, you would configure the following:

```
serverCharMapFile = "/etc/opt/cifsclient/unitables/unimapShiftJIS.cfg";
clientCharMapFile = "/etc/opt/cifsclient/unitables/unimapShiftJIS.cfg";
```

3. HP recommends that no other configuration modification be made to this file.

Because of limitations of the Windows file system, however, two configuration settings must be carefully considered for your operating environment: *execMapping* and *caseSensitive*. These are discussed in Chapter 5. Also, see the discussion of case sensitivity in the CIFS/9000 Server manual in the section “Other Samba Issues.”

NOTE As of this revision, Samba does not support Unicode. Windows NT and Windows 2000 use Unicode by default.

Step 4: Starting and Stopping the CIFS/9000 Client Daemon

Use the *cifsclient* command to start and stop the CIFS/9000 client.

The syntax is:

/opt/cifsclient/bin/cifsclient <start | stop>

The default, when no arguments are used, is to start the daemon. If the CIFS/9000 client is already running when you execute the command, you will get a message indicating it is already up.

Use the *stop* option of the *cifsclient* command to stop the CIFS/9000 Client.

In this case, the script tries to unmount all of the shares before it stops the daemon. If there is a problem with the unmounting, the script does not stop the CIFS/9000 Client.

Using the CIFS/9000 Client

This section presents a "quick start" overview of how the CIFS/9000 Client can be used. The basic procedure is (1) start the daemon, (2) mount shared directories, (3) login to CIFS Servers. These steps and some useful tips follow:

1. Starting the daemon.

   Normally the system administrator, logged in as *root*, would enter this command at system startup.

   ```
   $ cifsclient start
   Starting CIFS Client daemon 'cifsclientd' ... done;
     process id = 1911
   ```

   To check status at any time:

   ```
   $ cifsclient status
   ```

   ```
   path:     /opt/cifsclient/sbin/cifsclientd
   version:  FILESET HP CIFS CLIENT: Version: A.01.02
   cksum:    2843185805
   Status:   CIFS Client daemon is up; process id 1911,
             started Apr 13.
   ```

   You can also configure your system to start the CIFS Client automatically at bootup by editing the file */etc/rc.config.d/cifsclient* such that the run flag is set to 1: *RUN_CIFSCLIENT=1*. There must be no spaces on either side of the equal sign.

2. Mounting and unmounting shares on a CIFS server.

   This must be done by *root*. Directories mounted by the CIFS/9000 Client must first be configured as "shares" on the CIFS/9000 Server.

   In this example, the share *source*, configured as a "share" on the CIFS/9000 Server is mounted by the CIFS Client using the directory */home/devl/source* as the mount point. The directory used as the mount point must already exist.

   ```
   $ mount -F cifs buildsys:/source /home/devl/source
   ```

   To unmount:

   ```
   $ umount /home/devl/source
   ```

3. Accessing the shared directory via the mount point on the Client.

   The CIFS/9000 Client allows access to mounted directories only on a per-user basis. Therefore, each user must first be authenticated by the CIFS/9000 Server. This is accomplished through the *cifslogin* command.

   In this example, the share *source* has been mounted by the system administrator. The *root* user on the Client wants to access the shared directory on *buildsys*. This is first attempted by changing directories to the mount point, but without first logging into the server (this fails). Then, by logging into *buildsys* with the *cifslogin* command, the user is authenticated by *buildsys* and can access its shared *source* directory through the CIFS Client's mount point. Note that the user name used to login to the CIFS Server can be different than the current login name at the Client. The account and password pair used in *cifslogin* must exist on the system that does the authentication. Further, if the CIFS Server is an HP-UX system, all users on the Client that access the Server should have the same *uid* on both systems, so that file ownership is consistent.

```
$ whoami
root
cd /home/dev1/source
sh: /home/dev1/source: not found
```

   This fails because the user has not yet logged into the CIFS Server *buildsys*.

```
$cifslogin buildsys root
Remote user root's password: *****
```

   This succeeds. To verify the results.

```
$ cifslist -A

=========================================================
Server buildsys:
=========================================================
Remote Username: root          Local Username: root

Share: \\BUILDSYS\source
       rw /home/dev1/source

$ cd /home/dev1/source
$ _
```

Normal users (non-*root*) gain access to CIFS mounts in the same manner. Using the example above (*source* is mounted and root is authenticated on *buildsys*), a user named *lucy* accesses the mount as follows:

```
$ cifslogin buildsys lucy
Remote user lucy's password: *****
```

Verify results:

```
$ cifslist -A

============================================================
server buildsys:
============================================================
Remote Username: root               Local Username: root
Remote Username: lucy                Local Username: lucy

Share: \\BUILDSYS\source
        rw /home/devl/source
```

More on Mounting CIFS Filesystems

In addition to the mount command discussed in the previous section, which was used to explicitly create a single mount, there are other methods to manage the mounting of CIFS filesystems. See the reference for mount_cifs and umount_cifs in Chapter 4 for syntax details not contained in this section.

Using */etc/fstab*

By creating entries in */etc/fstab* you can mount CIFS filesystems automatically at boot time, or mount multiple CIFS filesystems, on one or more CIFS Servers, with a single command entered manually. The format for such entries is:

```
server:/share mount_point cifs defaults 0 0
```

Then, to mount all CIFS entries in */etc/fstab* manually, enter:

```
$ mount -aF cifs
```

To unmount all currently-mounted CIFS filesystems, enter:

```
$ umount -aF cifs
```

These commands will occur automatically, at bootup and shutdown respectively, if the system is configured to start the CIFS Client at bootup, as explained above.

NOTE Automounting a CIFS filesystem using the HP ONC+ Autofs service is NOT supported at this time. Attempting to mount CIFS filesystems via Autofs may cause hung processes to remain in the system process table and may make the mount point directory inaccessable.

How to Mount and Login in One Step

The root user has the option to mount a CIFS filesystem and login to the CIFS Server in one step, obviating the need to explicitly issue the *cifslogin* command. Using the names from the examples above:

```
$ mount -F cifs -o username=x,password=y buildsys:/source home/dev1/source
```

where *x* and *y* are the name and password pair recognized by the server.

Depricated mount and unmount commands

The *cifsmount* and *cifsumount* commands provide equivalent functionality to *mount* and *umount*, but their use is discouraged. They require different syntax and may not be available in future releases of the CIFS/9000 Client. *man* pages for these commands are provided in Chapter 3 of this manual for historical reasons.

CIFS/9000 Client Files and Directories

This section lists the important files that comprise the CIFS/9000 Client.

Table 2-1 CIFS/9000 Client Files and Directories

File/Directory	Description
/opt/cifsclient/	Base directory for all CIFS Client core files and administrative files.
/opt/cifsclient/bin/	CIFS Binaries.
cifsmount	For mounting CIFS Shares from CIFS Servers. Can only be used by root user
cifsumount	For un-mounting CIFS shares. Can only be used by root user.
cifslogin	For ordinary users to use the CIFS shares (already mounted), they should first login to the CIFS domain/machine with their username and password (according to CIFS configuration).
cifslogout	User logout from the CIFS domain. Cannot use the mounted shares in the CIFS domain.
cifslist	To list the mounted shares on the Client.
cifsclient	Start/Stop script for CIFS Client. Please refer to “Step 4: Starting and Stopping CIFS Client" for more details on this script.
/opt/cifsclient/pam	CIFS/9000 PAM files

Table 2-1 CIFS/9000 Client Files and Directories (Continued)

File/Directory	Description
/opt/cifsclient/sbin	CIFS Clients for use by the administrator or root user. Example: CIFS Client daemon is contained in this directory.
/etc/opt/cifsclient	Directory for CIFS Client configuration and localization files.
cifsclient.cfg	Configuration file accessed by CIFS Client daemon
cifsclient.cfg.default	Default configuration file. Should be copied as *cifsclient.cfg* for your use. Do not modify.
/etc/opt/cifsclient/unitables	Character-mapping tables for internationalized clients.
pam/smb.conf	PAM configuration file. May need to modify according to your needs. Refer to "Chapter 6: PAM NTLM" for more details on this file.
pam/smb.conf.default	Default PAM file. Should be copied as *pam/smb.conf* for your use. Do not modify.
/var/opt/cifsclient	Directory for the CIFS Client log files, pid files and any temporary files created for client's own use.

3 Commandline Utilities

This chapter provides details for the CIFS Client Commandline Utilities.

The CIFS/9000 Client software package consists of the following programs:

cifsclient	The command to use to stop and start the CIFS client.
cifsmount	The command to mount a directory from a remote server.
cifslogin	This command authenticates you to the remote server. You may then use any shares already mounted by other users.
cifsumount	This is the opposite of *cifsmount*. It removes the local mountpoint and disconnects it from the server if it is not mounted somewhere else.
cifslogout	This is the opposite of *cifslogin*. You cannot use any shares from the specified server after logging out.
cifslist	List connected servers, mountpoints, mounted shares, etc.
mount	mounts the CFIS filesystem.
umount	unmounts the CIFS filesystem

Each of the utilities described above also accepts the options *-h* and *-v* if given as the only parameter. The option *-h* prints a short help to standard error and the option *-v* prints the current version numbers to standard output.

NOTE

The sequence of arguments to the CIFS Client command-line utilities is significant, and it is reversed from conventional Unix commands. For example, in:

cifslogin server -U user

the argument pair *-U user* must appear after the argument server.

cifsclient

Synopsis

```
cifsclient start | stop | restart | status | ver [-v] [-x]
cifsclient force_umount moutpoint [ . . . ]
```

Description

cifsclient is a general-purpose shell script used to control global operations of the CIFS/9000 client. In its first form, it manages the CIFS client daemon, *cifsclientd*. The second form is used only for recovery from serious network error conditions. Entering "cifsclient" without options is equivalent to "cifsclient start".

To configure your system to start the CIFS Client on boot-up, see item one under "Using the CIFS/9000 Client" in chapter two of this manual.

Options

start	Start the CIFS client; the daemon forks and goes to the background immediately after startup. To configure your system to start the CIFS client on boot-up, see item 1 in Chapter 2 "Using the CIFS/9000 Client."
stop	Unmount all CIFS filesystems and stop the client. If the current working directory of any session on the CIFS client host is on a mounted CIFS filesystem, the unmount will fail with a "Device busy" error and the client will not shut down. Except in cases of error conditions (see the force_umount option below), the stop option should be used to shut down cleanly, rather than killing the CIFS client daemon directly.
restart	Equivalent to "cifsclient stop; sleep 1; cifsclient start".
status	Shows general information on the status of the daemon. Sample output is shown in Chapter 2 "Using the CIFS/9000 Client."

`ver`	Displays the version of the CIFS/9000 client product as stored in the host system's Installed Product Database. The `ver` argument recognizes the following options: `-v` Verbose; in addition to the product version, `what(1)` strings are displayed for all CIFS client binaries, scripts, and configuration files. `-x` Extra version information; in addition to the information displayed by `-v`, `-x` shows the CVS revision of each binary's component source file. This option is useful only for supported personnel.
force_umount	This option is intended to be used only in the case of serious errors. Its purpose is to force the host system to unmount CIFS filesystems that cannot be unmounted by the CIFS client. This condition can result from certain types of network errors. If the CIFS client daemon is up, you must kill the *cifsclientd* process before using `force_umount`.

Files

/etc/opt/cifsclient/cifsclient.cfg

This file contains run-time configuration options for the CIFS/9000 Client. For detailed information see Chapter 5, “Configuration File”.

See Also

cifsmount, *cifslogin*, *cifsumount*, *cifslogout*, *cifslist*

cifsmount

You can use the *mount* command to execute the *cifsmount* command. Both commands are shown below.

Synopsis

`cifsmount //<server>/<share> <mountpoint> [<options>]`

Description

The *cifsmount* command is used to mount remote shares on the local filesystem. It mounts the share <*share*> from server <*server*> in the local filesystem at <*mountpoint*>. The mountpoint must exist. The user is prompted for a password and the program uses the combination username/password to log in to the server. If the current user is already logged in to the given server, the password prompt is skipped. You can use the option -*N* to suppress password prompting.

NOTE HP does not support the SSL option.

Options

`-c <clientname>`

Set Netbios name of client. CIFS is based on Netbios. Netbios requires that valid Netbios computer names are supplied during the connection establishment for the client and the server. The client name is usually taken from the hostname of your computer. If this does not work or if your computer's Netbios name is different, you may supply the value to be used with this parameter. This parameter is ignored if the server is already connected.

`-I <IP number>`

IP address of server. By default, the hostname of the server is taken from the server specification of the share. This must also be the Netbios host name of the server, if the server enforces correct netbios names. CIFS/9000 Client uses DNS instead of Netbios to resolve server names to IP addresses. If the DNS name of the server is different from the Netbios name, you may supply the DNS name or the server's IP address with this parameter. It is ignored if the server is already connected.

`-p <portnumber>`

Set connection port. Netbios connections are usually made on port 139. If you want to connect on a different port, you can supply a decimal port number with this parameter. This parameter is ignored if the server is already connected.

`-r` Mount as read-only filesystem.

`-U <username>`

Username sent to server. By default, CIFS/9000 Client accesses the server under the same user name as the login name of the user that issues the cifsmount command. If you have a different user name at the server, you may use this option to set that name. It is ignored if you are already logged in at the server.

`-P <password>`

Password given in commandline. Use this option only if necessary, because all commandline parameters may show up in the output of the ps command. It gives you the possibility to pass a dynamically generated password to the server. The password is ignored if the user is already logged in at the server.

`-S` Read the password from stdin. This option may be useful if you want to use cifsmount from a shell script or an other program. The *-P* option is insecure for this purpose because the unix command ps can show the commandline parameters of running processes.

`-N` Do not prompt for a password. This option may be used to avoid prompting for a password if the user does not have a password.

`-u` Enable plain text passwords. CIFS/9000 Client refuses to send passwords in plain text to the server by default because this is a security risk. There are tools available that sniff the network for plain text passwords. If you really must send the password in plain text (e.g. because your server does not allow password encryption), you can enable it with this option. It is ignored if the user is already logged in at the server.

`-f` Force mount. When this option is used, the mount is even done if the server is not responding. No requests are sent to the server. Consequently, none of the parameters can be checked for validity.

`-s` Save mount and password in database. Do not use unless you understand the security implications. CIFS/9000 Client can maintain a database of mounts, usernames and passwords. This database is used at startup to re-establish stored mounts and to log in users on demand, even if the user is not logged in at the client. This option may be useful for automounting and to run programs by cron that have no possibility to ask the user for a password. Passwords are stored in CIFS/9000 Client's user database file. It is possible to get the CIFS/9000 hash values of the passwords (which is functionally equivalent to the passwords themselves) out of this file, although the file itself is not sufficient. You can use this option safely only if you are the only one who has physical or root access to your machine or if you trust everyone who has this access. CIFS/9000 Client does not store unencrypted passwords in the user database. If your server does not support encrypted passwords, you cannot use this option.

Examples

The following command mounts the share entiredisk from the server bigserver at the local mountpoint */mounts/bigserver* and stores the mount and the user/password combination in the user database.

`cifsmount` *//bigserver/entiredisk /mounts/bigserver -s*

Files

Mounts, usernames and passwords are stored encrypted in CIFS/9000 Client's user database file. The path to the user database file can be configured in the CIFS/9000 Client configuration file. The default path is

/var/opt/cifsclient/cifsclient.udb

See Also

cifslogin, cifsumount, cifslogout, cifslist

cifslogin

Synopsis

`cifslogin <servername> [<username>] [<options>]`

Description

The *cifslogin* command is used to authenticate additional users at a server. Only authenticated users may access mounted files. Each user accesses the file at the server with his or her privilege status at that server. Because there must be a one to one (many to one) mapping from local users to remote user names, every user can log in only once at a given server. By default, *cifslogin* sends the user's login name to the server. If this is not desired, the username can be given in the commandline.

Options

`-c <clientname>`

Set Netbios name of client. CIFS/9000 is based on Netbios. Netbios requires that valid Netbios computer names are supplied during the connection establishment for the client and the server. The client name is usually taken from the hostname of your computer. If this does not work or if your computer's Netbios name is different, you may supply the value to be used with this parameter. This parameter is ignored if the server is already connected.

`-I <IP number>`

IP address of server. By default, the hostname of the server is taken from the server specification of the share. This must also be the Netbios host name of the server, if the server enforces correct Netbios names. CIFS/9000 Client uses DNS instead of Netbios to resolve server names to IP addresses. If the DNS name of the server is different from the Netbios name, you

may supply the DNS name or the server's IP address with this parameter. It is ignored if the server is already connected.

`-p <portnumber>`

Set connection port. Netbios connections are usually made on port 139. If you want to connect on a different port, you can supply a decimal port number with this parameter. This parameter is ignored if the server is already connected.

`-P <password>`

Password given in commandline. Use this option only if you really have to, because all commandline parameters may show up in the output of the *ps* command. It gives you the possibility to pass a dynamically generated password to the server. The password is ignored if the user is already logged in at the server.

`-S` Read the password from stdin. This option may be useful if you want to use *cifslogin* from a shell script or an other program. The *-P* option is insecure for this purpose because the unix command *ps* can show the commandline parameters of running processes.

`-N` Do not prompt for a password. This option may be used to avoid prompting for a password if the user is already logged in at the server or if the user does not have a password.

`-u` Enable plain text passwords. CIFS/9000 Client refuses to send passwords in plain text to the server by default because this is a security risk. There are tools available that sniff the network for plain text passwords. If you really must send the password in plain text (e.g. because your server does not allow password encryption), you can enable it with this option. It is ignored if the user is already logged in at the server.

`-f` Force login. When this option is used, the login is even done when the server is not responding. No requests are sent to the server. Consequently, none of the parameters can be checked for validity.

`-s` Save password in database. Do not use unless you understand the security implications. This option CIFS/9000 Client can maintain a database of mounts, usernames and passwords. This database is used at startup to re-establish stored mounts and to log in users on demand, even if the user is not logged in at the client. This option may be useful for automounting and to run programs by cron that have no possibility to ask the user for a password. Passwords are stored in CIFS/9000 Client's user database file. It is possible to get the CIFS hash values of the passwords (which is functionally equivalent to the passwords themselves) out of this file, although the file itself is not sufficient. You can use this option safely only if you are the only one who has physical or root access to your machine or if you trust everyone who has this access. CIFS/9000 Client does not store unencrypted passwords in the user database. If your server does not support encrypted passwords, you can't use this option.

Examples

If local user steve has mounted a share from server *bigserver*. Local user bill has no access to the mounted files because he is not logged in at the server. Bill, who has an account on *bigserver* under his real name miller can do the following to gain access:

```
cifslogin bigserver miller
```

Bill will be prompted for a password and if it's correct, he will be given access to the share with the same privileges that the user miller has on *bigserver*.

Files

Usernames and passwords are stored encrypted in CIFS/9000 Client's user database file. The path to the user database file can be configured in CIFS/9000 Client's configuration file. The default path is

/var/opt/cifsclient/cifsclient.udb

See Also

cifsmount, cifsumount, cifslogout, cifslist

cifsumount

You can use the *umount* command to execute the *cifsumount* command. Both commands are shown below.

Synopsis

`cifsumount <mountpoint> [<options>]`

`cifsumount -a [<options>]`

Description

The *cifsumount* command is used to unmount any shares mounted with *cifsmount*. Shares can only be unmounted by the user that mounted the share at the given mountpoint or the superuser. The second variant (with the *-a* option) unmounts all mounts that are currently served.

Options

`-d`	Delete Mount from database. If the mount associated with <*mountpoint*> is stored in the user database, it is deleted from that database.
`-l`	Also log out all users from server, if no more shares are mounted from the server (default behaviour).
`-k`	Keep users logged in at the server, even if no shares are mounted.
`-f`	Force unmount: Avoid requests to the server (useful if server is down).

Files

Mounts, usernames and passwords are stored encrypted in CIFS/9000 Client's user database file. The path to the user database file can be configured in CIFS/9000 Client's configuration file. The default path is

/var/opt/cifsclient/cifsclient.udb

See Also

cifsmount, *cifslogin*, *cifslogout*, *cifslist*

cifslogout

Synopsis

`cifslogout <servername> [<options>]`

Description

The *cifslogout* command is used to log the user who uses the command out of the server specified. After issuing *cifslogout*, the user cannot access any files from that server unless he or she is still stored in the user database.

Options

`-d`	Delete password from database. If the user's password is stored in the user database, it is deleted from that database.

Files

Mounts, usernames and passwords are stored encrypted in CIFS/9000 Client's user database file. The path to the user database file can be configured in CIFS/9000 Client's configuration file. The default path is

/var/opt/cifsclient/cifsclient.udb

See Also

cifsmount, cifslogin, cifsumount, cifslist

cifslist

Synopsis

`cifslist -A` lists servers with shares and mountpoints

`cifslist -U` lists users in database

`cifslist -M` lists mounts in database

`cifslist -S` lists connected servers

`cifslist -s <server>` lists shares open at server

`cifslist -u <server>` lists users logged in at server

`cifslist -m <share>` lists mountpoints for share

Description

The *cifslist* command is used to view internal tables of CIFS/9000 Client.

mount_cifs, umount_cifs

Mount and unmount CIFS filesystems.

Synopsis

```
mount -F cifs [-ar] [-o option[,option...]] [server:/share mount_point]
umount -aF cifs | mount_point
```

Description

The *mount* command mounts file systems. Only a superuser can mount file systems. Other users can use *mount* to list mounted file systems. Use *cifslist -A* to view CIFS-specific mounts and user connections.

The *mount* command attaches *server:/share* to *mount_point*. *server* is a remote system. *share* is a directory on this remote system and *mount_point* is a directory on the local file tree. *mount_point* must already exist, and be given as an absolute path name. It will become the name of the root of the newly mounted file system.

If *mount* is invoked without any arguments, it lists all of the mounted file systems from the file system mount table, */etc/mnttab*.

The *umount* command unmounts currently-mounted file systems. Only a superuser can unmount file systems.

Options

`-F cifs`	Filesystem-specific identifier. Always required for mounting and unmount CIFS filesystems, except for the command form *umount moint_point*.
`-a`	Used with *mount*, mounts all CIFS filesystems that have entries in */etc/fstab*. Used with *umount*, unmounts all currently-mounted CIFS filesystems.
`-r`	Mount as read-only.
`-o`	This class of options is specified with the following syntax: -o keywrd[,keywrd...],keywrd=value[,keywrd=value...]

That is, some keywords are specified as keyword/value pairs, some are not -o options must be delimited by commas; no white space is allowed. For example:

-o ro,username=fulton,password=pokey

Following are the *-o* options to *mount* supported by the CIFS Client (keywords that require values are indicated by "keyword=*value*"):

nbname=*nbname* Set NetBios name of client. CIFS/9000 is based on NetBios. NetBios requires that valid NetBios computer names are supplied during the connection establishment for the client and the server. The client name is usually taken from the hostname of your computer. If this does not work or if your computer's NetBios name is different, you may supply the value to be used with this parameter. This parameter is ignored if the server is already connected.

ipaddr=*addr* IP address of server. By default, the hostname of the server is taken from the server specification of the share. This must also be the NetBios host name of the server, if the server enforces correct NetBios names. The CIFS/9000 Client uses DNS instead of NetBios to resolve server names to IP addresses. If the DNS name of the server is different from the NetBios name, you may supply the DNS name or the server's IP address with this parameter. It is ignored if the server is already connected.

port=*port* Set connection port. NetBios connections are usually made on port 139. If you want to connect on a different port, you can supply a decimal port number with this parameter. This parameter is ignored if the server is already connected.

ro Mount as read-only filesystem.

username=*name* Username sent to server. By default, the CIFS/9000 Client accesses the server under the same user name as the login name of the user. If you have a different user name at the server, you may use this option to set that name. It is ignored if you are already logged in at the server.

password=*passwd* Password given in commandline. Use this option only if you really have to, because all commandline parameters may show up in the output of the *ps* command. It makes it possible to pass a dynamically generated password to the server. The password is ignored if the user is already logged in at the server.

`plaintxt` Enable plain text passwords. The CIFS/9000 Client refuses to send passwords in plain text to the server by default because this is a security risk. There are tools available that sniff the network for plain text passwords. If you really must send the password in plain text (e.g. because your server does not allow password encryption), you can enable it with this option. It is ignored if the user is already logged in at the server.

`forcemnt` When this option is used, the mount is done even if the server is not responding. No requests are sent to the server. Consequently, none of the parameters can be checked for validity.

Files

/etc/mnttab table of mounted file systems.
/etc/fstab list of default parameters for each CIFS filesystem.

See Also

mount (1M), *umount*(1M), *cifslogin*, *cifsumount*, *cifslogout*, *cifslist*

4 Troubleshooting the CIFS/9000 Client

This chapter includes information about problems that you may encounter when using the CIFS/9000 client and explanations of error

messages that might occur with CIFS/9000 commands.

- Troubleshooting FAQs
- CIFS/9000 Client Error Messages

Troubleshooting FAQs

This section includes commonly asked questions about CIFS/9000.

How to Kill the Daemon with cifsclient stop

You should never kill the daemon process directly. Although CIFS/9000 tries to unmount all mounted shares, it may not be successful and the stale mounts will become unusable and cause problems. The correct way to do it is with *cifsclient stop*.

Refer to “Step 4, Starting and Stopping the Client” in chapter 2 in this manual for more detailed information about *cifsclient stop*.

What to Do if the Daemon Terminates

If the daemon terminates, all shares served by CIFS/9000 will immediately become unusable. Every access will hang until the NFS timeout (configured in the configuration file) elapses. You can probably get away without rebooting if you immediately terminate all processes using the mounts, change all current directories from within the mounts and then use the cifsclient force_umount <mountpoint> command to unmount the stale mounts. Report the event to HP Technical Support and describe how the problem can be reproduced.

CIFS/9000 Client Error Messages

This section contains information about CIFS/9000 Client error messages for the following commands:

- cifsclient
- cifsmount
- cifslogin
- cifsumount
- cifslogout
- cifslist

`userdb: cannot open file`

`/var/opt/cifsclient/cifsclient.udb`

cifsclient was unable to open the user database file for the reason given in the message. This is normal if CIFS/9000 is started the first time or if nothing has been stored in the user database so far.

`userdb: database file is incompatible`

cifsclient has found a user database file, but this file was written by a different version of CIFS/9000 Client. For security reasons, all versions of CIFS/9000 Client (even from different compiler runs) are incompatible in this respect.

`ipcclient: error connecting to daemon: ...`

The commandline utility was not able to connect to the CIFS/9000 Client daemon. The detailed unix error message is given. Most probably the daemon is not running.

`LOC: Server can't encrypt passwords, use option to override`

Your server does not support encrypted passwords. CIFS/9000 Client refuses to send passwords in clear text by default. You can override this with the *-u* option. You should be aware, however, that anyone with a network sniffer can read your unencrypted password on the network.

Almost every Unix-machine can be turned into a network sniffer. CIFS/9000 Client also refuses to store unencrypted passwords in the user database.

`error: DOS: Access denied`

The username/password pair you supplied was not accepted by the server. You may try to supply the username explicitly with the *-U* option.

5 Configuration File

The default configuration file should work without modifications. Do not modify the configuration file unless you are sure you know what you are doing.

The configuration file is parsed by the CIFS/9000 Client daemon at startup and when edited. Although it is re-read by the running daemon, not all configuration changes will work immediately. Most options are read into internal variables when they are used. The server configuration, for instance, is transferred into internal structures when a connection to the server is opened. Therefore, if a change to the server configuration should become active, you must first unmount all shares and log out all users from that server.

NOTE SSL Options are not supported in CIFS/9000.

General Structure

Configuration files are built from the following simple syntactic structures:

- remarks
- strings
- arrays
- dictionaries

Strings, arrays and dictionaries are classified by the generic term "property".

Remarks can be written in three forms:

```
/* remark */
```

as in C,

```
// remark to end of line
```

as in C++ or Objective-C and

```
# remark to end of line
```

as in shell scripts.

Strings are sequences of alphanumeric characters, including the underscore. If a string should consist of other characters like spaces, it must be quoted in double quotes. Within double quotes, the same escape-sequences as in C strings can be used. There is no separate syntax for numeric arguments. Numeric arguments are regarded as strings and converted when used.

Arrays are ordered lists of other properties. An array is delimited by parentheses and the properties constituting the array may be separated by commas. The following example is an array consisting of several string elements:

```
(1, 2, 3, hello, "how are you")
```

Dictionaries are unordered lists of named properties. These lists are delimited by curly braces. Each dictionary entry consists of a left hand side (key), which must be a string, an equal sign and a right hand side (value) which may be any property. Entries may be separated by

semicolons. The following is an example of a dictionary consisting of three entries named *property1* to *property3* where the first one has a string value, the second an array value and the third a dictionary value:

```
{
      property1 = "value of property1";
      property2 = (value, of, property2);
      property3 = {
          firstWord = value;
          secondWord = of;
          thirdWord = property3;
      };
  }
```

The configuration file itself is a dictionary (the surrounding curly braces are optional because other properties are not allowed). The keys at the top level are the names of the configuration variables.

Properties that have been parsed as strings may be interpreted in one of the following ways:

- string
- number
- enumeration
- boolean

String needs no further explanation. Numbers are interpreted in decimal, unless they are prefixed with 0 (meaning octal) or 0x (meaning hexadecimal). Enumerations are strings from a predefined set of strings. And boolean variables are a special case of enumeration where the set consists of the strings *yes* and *no*.

Configuration Variables

The following is a list of all variables that may be configured at the top level:

logLevels The value of this variable is an array enumerating all logging modes that are active. A logging mode is a string out of the following set:

`info`

[0] Logging of informational messages. Should be turned on.

`error`

[1] Logging error messages. Should really be turned on.

`debug`

[2] General debug messages. Useful only during debugging.

`resource`

[3] Messages about allocation and deallocation of objects. Useful only during debugging.

`netbiosError`

[4] Logging error messages from the Netbios layer. Should be turned on, unless too many errors occur. This is separated from the general error logging because not all of Netbios is implemented in CIFS/9000 Client and the unimplemented features result in Netbios error messages.

`netbiosDebug`

[5] Debug messages from the Netbios layer. Useful only during debugging.

`netbiosTrace`

[6] This option generates hex-dumps of all outgoing and incoming Netbios traffic. This is very useful during debugging but should really be turned off for normal operation.

`nfsTrace`

[7] This class of logging messages provides detailed information about all NFS requests done by the kernel and the respective return values. It is very useful for debugging the NFS part but should really be turned off for normal operation.

`rare`

[8] Logging of rare conditions. Useful only during debugging.

`cacheDebug`

[9] Debugging of the cache's operation. Useful only during debugging.

`cifsTrace`

[10] Logging of all CIFS commands issued and the respective return values. Very useful together with netbiosTrace for debugging, but should really be turned off during normal operation.

`oplock`

[11] Debugging of opportunistic lock mechanism. Useful only during debugging.

`warn`

[12] Warnings of any kind, mostly used by the configuration file parser. Should be turned on.

`smbSequence`

[13] Debugging messages about the order of CIFS requests and the respective messages. Useful only during debugging.

`debugAttributes`

[13] Debugging of file attribute routines. Useful only during debugging.

The numbers in square brackets which precede the descriptions are used to denote messages of the respective logging mode in the logging output.

cfgParseInterval CIFS/9000 Client can re-parse the configuration file while running. For this feature to work, CIFS/9000 Client must poll the file regularly. The variable *cfgParseInterval* defines the time of this poll cycle in milliseconds. If it is set to 0, the file is only parsed once during startup. The default is 0.

sockMode
sockOwner
sockGroup The file access mode and ownership for the Unix domain socket that is used for communication between the CIFS/9000 Client daemon and the commandline utilities. The access mode may be given in octal notation, if prefixed with a leading 0, in hexadecimal notation if prefixed with a leading 0x or in decimal notation if not prefixed with any of the above. Owner and group may be given by name or as numeric id. Do not set these values to anything other than *mode=0600* and *owner=root* unless you really know what you are doing. The file access modes of this unix domain socket are used to provide secure authentication of the user that requests a service to the daemon. If these variables are not configured from the file, they default to the correct values.

runAsUser The CIFS/9000 Client daemon must be started as root. To improve security, it switches to a different user-id if root privileges are not needed. The user-id that shall be used for this purpose can be configured with this value. It may be either a user name or a numeric id.

pidFile CIFS/9000 Client can maintain a file with the process id of the daemon, if desired. If this variable is defined, it is interpreted as the path of the file where the pid should be stored. If it is not defined, no such file is created.

databaseFile This variable configures the path to the user database file. It stores passwords, mounts and the registration key. The default is */var/opt/cifsclient/cifsclient.udb*.

allowSaving This boolean variable defines whether user passwords and mounts may be stored in the user database. Setting it to *no* disables storing. The default is *yes*.

caseConvertFile This variable configures the path to the case conversion table. This file defines the mapping to upper and lower case for all unicode characters. The default is to use no table file and retain the default ISO 8859-1 mapping. A mapping file derived from the Unicode standard is part of the CIFS/9000 Client distribution. You can find it at *unitables/unicase.cfg*.

serverCharMapFile

This variable configures the path to the character mapping file for the server. This file is only used when client and server do not agree on using Unicode. It defines the mapping from the internal Unicode representation to the ASCII strings sent to the server (and vice versa). The default is a "codepage 437" mapping, which is the US-Latin DOS character set. Mapping files for various character sets are distributed with CIFS/9000 Client in the directory *unitables*.

clientCharMapFile

This variable configures the path to the character mapping file for the client. This file defines the mapping from internal Unicode representation to the ASCII strings seen at the client. Together with the *serverCharMapFile*, any conversions between server and client character code can be accomplished. These tables can be used to compensate for vendor specific character sets and to cope with various national character sets such as JIS and ShiftJIS for Kanji etc. The default is an ISO 8859-1 mapping.

uniTableCompressBlocks

This integer variable customizes the compression of the Unicode table. A higher value reduces conversion speed but improves memory efficiency. Values higher than the number of contiguous unused code blocks have no effect. The default is 3.

nfsSockRxBuf This integer variable sets the receive buffer size of the socket used to communicate with the kernel. If the value given is out of the acceptable range for your

machine, CIFS/9000 Client automatically limits the range. Increase the buffer size if you have extremely slow writes.

nfsSockTxBuf This integer variable sets the transmit buffer size of the socket used to communicate with the kernel. It should not be necessary to set an explicit buffer size.

nfsTransferSize This integer variable defines the maximum block size used in data transfer between the kernel and CIFS/9000 Client. The maximum allowed value is 8k (8192). It may be necessary to reduce the value if the NFS socket has frequent overflows, as it may be the case with AIX 3.x. It's useful to use only powers of 2 as block sizes. The default is 8192.

scopeID This string variable defines the Netbios name scope of the client. If it is not defined, no scope ID is used. If you don't know what a scope ID is, you don't need one.

defaultServer The baroque structure of CIFS has its mirror in the multitude of configuration options for CIFS connections. This variable defines a default behavior which can be overridden by specific configurations for each server. The value is a dictionary with the following keys:

localNetbiosName

This entry can be used to set the Netbios name for the client that is sent to the server.

mtabName This string variable defines the hostname that is used in mount table entries. Solaris may be very picky about this name. If the given host exists and does not provide NFS (or RPC) services, logging in may take a long time. If this variable is not defined, it defaults to the *hostname* (or *IP address*, if specified) of the server.

connectTimeout	This integer variable defines the maximum time in milliseconds that is waited for a connection to succeed. You probably have to increase the time if you are on a slow network. The default is 2000ms (2 seconds).
requestTimeout	This integer variable defines the maximum time in milliseconds a server response may take (if the connection is already established). The default is 60000ms (60 seconds).
nfsTimeout	This integer variable defines the initial timeout in 1/10 seconds that is used by the kernel when it requests data from CIFS/9000 Client. This value is doubled on each retry. Together with *nfsRetransmit*, this defines the absolute timeout for NFS requests. A value of 50 (5 seconds) avoids frequent retries of already running (slow) requests and ensures a total timeout of about 2 minutes. This should be sufficient even for the slowest devices and links. If you use a jukebox, it may also be necessary to increase *requestTimeout*.
nfsRetransmit	This integer variable defines the number of retries the kernel attempts when CIFS/9000 Client does not reply in time. The timeout starts with *nfsTimeout* and is doubled on each retry. Retransmissions should not be necessary, because CIFS/9000 Client should not lose any requests. However, if your system's NFS client puts high loads on NFS servers and has small maximum socket buffer sizes, requests can get lost due to buffer overflows. A value of 5 (which is also the default) should be a good choice. You may want to experiment

with *nfsTimeout* to get the optimum performance even with frequent buffer overflows.

nfsAttributeCaching
: A boolean variable that can enable file-attribute caching by NFS, effectively overriding the CIFS client's attribute cache). This improves performance of certain types of operations (such as creating *tar*(1) archives of large numbers of files residing on mounted CIFS filesystems), by reducing the number of "get attribute" calls sent over the network. The deault setting is "no".

sslRequireEncryption
: If this boolean variable is set to *yes*, connections to non-SSL servers are refused. Otherwise the server determines whether SSL (Secure Socket Layer) is used or not.

sslVersion
: SSL is an evolving standard and there are several versions around. This enumeration variable defines the version(s) that will be used. It can be set to *ssl2*, *ssl3*,*ssl2or3* or tls1. The default is to automatically negotiate version 2 or 3 (*ssl2or3*).

sslCompatibility
: SSLeay can be configured to behave compatibly to bugs in other SSL implementations. This is useful for interoperability, although it contradicts strict adherence to the standard. Currently, all SSL CIFS servers use SSLeay, too, and there's no need to be compatible to something else. This boolean variable switches on compatibility mode if set to *yes*.

sslCertFile	SSL servers can (and probably should) be configured to require a certificate from the client. If the client needs a certificate, this string variable points to the file containing the certificate in PEM format.
sslKeyFile	A certificate is specific for a key. If you use a certificate, you must also specify the key. If the key is not already contained in the certificate file, this variable should contain the path of a file with the key in PEM encoded format.
sslServerCert	This variable defines whether the server must present a valid certificate during negotiation. It must be set to a dictionary containing the keys *required* and *allowOverride*. The entry *required* has a boolean value which determines whether a server certificate is required or not and the entry *allowOverride* must also be set to a boolean value which determines whether the value set for *required* can be overridden from the commandline of *cifsmount*.
sslCACertDir	If a server certificate is required, there must be a way to verify it. This string variable points to a directory containing the certificates of all trusted certification authorities (CAs) in hashed format (file names are hash values of the CA names).
sslCACertFile	Alternatively (or in addition) to defining a directory with CA certificates, you may also specify a single file containing all CA certificates in PEM format. This string variable contains the path of that file.

sslCiphers	This string variable can be used to set the preferred ciphers (should not be necessary because the ciphers are negotiated during connection setup).
lookupStrategy	This configuration variable needs some explanation. As you probably know, CIFS/9000 Client maps between NFS requests and SMB/CIFS requests. On the NFS side, files are referenced by unique identifiers, the so called NFS file handles. On the CIFS side, files are referenced simply by their path. CIFS/9000 Client must be able to determine the path given an NFS file handle. There are two strategies available to do this: • pseudoInode This strategy derives the NFS file handle as a hash value from the path. The hash is chosen in a way that makes efficient lookups possible, as long as the depth of the file in the directory hierarchy is lower than 27. The advantage of this strategy is the low memory consumption: Files can be looked up on demand, nothing has to be stored. The main disadvantage is that NFS file handles change when files are renamed. This leads to a conflict with Unix semantics when open files are renamed: After renaming, the handle of the open file is stale and the file can not be accessed without reopening. It also conflicts with a bug in the caching code of the Solaris NFS client

where the writeback occurs only after closing the file, not during closing the file.

- database

  In this strategy all NFS file handle to file path relations are stored in an internal database. This is the most secure and most compatible approach. The disadvantage is that all this information must be kept in memory. CIFS/9000 Client needs about 500kB more real memory and about 10MB more virtual memory per share that uses this strategy.

The *database* strategy is the default.

caseSensitive — This is a boolean variable (possible values *yes* or *no*) which specifies whether filenames on the server are case sensitive. By default they are case sensitive in order to be consistent with the Unix file system. If you use a case mapping different from *none* (see next parameter), you must set this parameter to *no*.

caseMapping — This variable (of type enumeration) defines whether file names are mapped to all upper case (*upper*), all lower case (*lower*) or preserved as they are on the server (*none*).

capitalizeShares — This boolean variable defines whether share names are converted to all uppercase characters before a connection is attempted. Share names should be case insensitive, but Windows 95 does not accept lowercase names. If this option occurs

in section *serverClasses*, it can override a *no* to a *yes*, but not a *yes* to a *no*. The default is *yes*.

useUnicode — This boolean variable specifies whether CIFS/9000 Client will use Unicode if the server supports it.

domain — This string variable defines the domain name the client sends to the server. If undefined, it defaults to an empty string which should be OK for all known servers.

alwaysEncryptData

If this boolean variable is set to *yes*, only SSL (Secure Socket Layer) connections with the server are accepted. If set to *no*, SSL is negotiated with the server.

guestUser — The *guestUser* configuration solves the following problem: each Unix user must be logged in at the server (be mapped to a CIFS username/password pair) in order to access anything, even if the share is public. It may be impractical to log in each user if there's a large number of Unix users who want to access a public share where access permissions are not important. If you define a *guestUser*, all Unix users that are not logged in are treated as if they were the given Unix user. The Unix user named in *guestUser* should be logged in, of course, e.g. with the *-s* option to *cifsmount* or *cifslogin*.

fakeMountpointDate

If this boolean variable is *yes*, the modification and access times of the mount point always read the current time. This is useful for servers that

return bogus values for the modification dates of root directories, such as Windows NT or Windows 95. The default is *no*.

execMapping — This enumeration variable is useful for files stored on Windows servers. It defines which DOS attribute whould be mapped to the Unix *execute* permission. The following keywords are valid: *archive*, *system*, *hidden*, *on*, or *off*. Default is *on*. A side-effect of *execMapping* is that if the configured attribute is set on the NT server, the file will be listed on the Unix Client with the execute bit set for all users (*owner*, *group*, and *other*).

WARNING

If you plan to store Unix executables on an NT server and invoke them on a Unix Client, then the default setting *execMapping = on* is required. In this case, as seen by the Unix Client, the execute bit is set on *all* file listings from the Windows server. Using *execMapping = on* will not affect the attributes of files on CIFS/9000 Servers; those will still behave like normal Unix files.

---

execInvert — When this boolean variable is *yes*, the execute bit (as derived with the *execMapping*-setting) is inverted.

dirDefaultLinks — If the server does not supply a number of hard-links for directories, this number is used. The value defaults to 2, if not specified. Some implementations of the Unix utility

find determine whether recursion is necessary or not from the link count. If your find uses this optimization, you may want to fake a high number of links for directories. Alternatively you can switch off the optimization with a commandline switch to find.

enableFakeLinks If this boolean variable is set to *yes*, CIFS/9000 Client can do softlinks on Windows-servers. These softlinks can be used by CIFS/9000 Client clients only, of course. On the Windows server they look like ordinary files with special attributes set (system and hidden attributes, if you have not modified the configuration).

linkModeMask, linkMode

These two integer variables define the file attributes that are used to distinguish faked softlinks from ordinary files. *linkModeMask* is 7 by default, which means that the attributes *read-only*, *hidden* and *system* are taken into account. *linkMode* defines the actual state that these attributes must have. It's 6 by default, which means that *hidden* and *system* must be set, but not *read-only*. The configuration value is calculated as sum of the following components:

Table 5-1

1	read-only	2	hidden	4	system	32	archive

linksAreUnicode If this boolean variable is set to *yes*, CIFS/9000 Client stores faked links in Unicode format on the server. This is incompatible with the CygWin32 format for symbolic links, but allows

lossless storage of client paths. If it is set to *no*, symbolic links are more or less compatible to those of CygWin32 on Windows, but a conversion to the server character set is performed. Regardless of this variable, CIFS/9000 Client can read symbolic link files in both formats.

attributesCacheTime

File attributes are cached for this amount of time (in milliseconds).

dirCacheTime — Directory contents are cached for this amount of time (in milliseconds).

maxCachedFiles — This is the maximum number of file objects that are held as cache of NFS file handles. If an NFS file handle is requested which is not in the cache, it must be looked up recursively, which may result in a notable performance loss. Recursive lookups are logged as rare events.

maxOpenFiles — This is the maximum number of files that will be kept open at the server.

dataCacheSize — This is the size of the data cache that is allocated for open files in bytes. The value is rounded to a multiple of the cache's page size, which is derived from the maximum transferable size. The page size will always be a power of two.

closeDelay — This variable defines the time a file is kept open when it is not used. The value is a dictionary with the following keys:

exclusiveLock

The keep-open time in milliseconds if an exclusive *oplock* has been acquired.

batchLock

The keep-open time in milliseconds if a batch *oplock* has been acquired.

noLock

The keep-open time in milliseconds if no lock has been granted.

dataCacheTimeNoLock

If no *oplock* has been granted, no caching should be done. This might result in bad performance on servers that do not support oplocks. This value sets a cache-valid time (in milliseconds) that is used if no *oplock* was granted.

readAhead — This variable defines the number of cache pages to read ahead. It is a dictionary with the following keys:

lock

The number of pages to read ahead if an *oplock* was granted.

noLock

The number of pages to read ahead of no *oplock* was granted.

useWriteBack — This variable defines whether cache write back techniques should be used. Write back is insecure (in terms of error recovery) if used with NFS2, but it may increase performance notably. The value is a dictionary with the following keys:

lock

Boolean value which configures whether write back should be used when an *oplock* has been granted.

noLock

	Boolean value which configures whether write back should be used when no *oplock* has been granted. If you care about reliability, always leave these options off. This configuration variable is also passed to the server. There are server/OS combinations (notably Samba/Linux) which become very slow in write through mode. You may want to configure write back for these.
requestOplock	This boolean variable defines whether oplocks should be requested from the server. It should be set to *no* for Windows95 machines because they grant an *oplock* although there is no support for it.
closeForSetattr	This boolean variable defines whether files should be closed before attributes (write protection, modification dates) are changed. This is very useful for Windows 95 servers because these servers can't set the attributes of open files. However, with this feature enabled, the unix semantics mapping does not work completely. The default is *no*.
disableSmbs	Not every server supports every SMB command equally well. In fact, many commands are unusable on certain server types. The value of this variable is an array which enumerates the SMB commands that should not be used. The respective commands will be replaced by a workaround automatically. The enumeration constants may be taken from the following set: `getattrFind`

Suppresses the use of the *trans2/findfirst2* command for reading file attributes. *Trans2/findfirst2* is the best way to query attributes, so only disable it if you need to.

`getattrTrans2QueryPath`

Suppresses the use of the *trans2/query_pathinfo* command for reading file attributes. *Trans2/query_pathinfo* seems to be broken on Windows 95.

`attrUnix`

Disables the Unix extensions for file attributes.

`setattrTrans2SetFile`

Suppresses the command *trans2/setfileinfo* to be used for setting file attributes. This SMB command does not work properly on NT.

`setattrTrans2SetPath`

Suppresses the command *trans2/setpathinfo* to be used for setting file attributes. This SMB command does not work properly on NT.

`setattrSetFile2`

Suppresses the use of *SET_INFORMATION2* for setting attributes.

`setattrCoreWithTime`

Suppresses the use of the core *SET_INFORMATION* command for setting modification dates.

`createOpenX`

Suppresses the use of *SMB_COM_OPEN_ANDX* for creating files.

`openOpenX`

Suppresses the use of *SMB_COM_OPEN_ANDX* for opening files.

`readReadX`

Suppresses the use of *SMB_COM_READ_ANDX* for reading files.

`readOpenRead`

Suppresses the use of *SMB_COM_OPEN_ANDX* batched with *SMB_COM_READ_ANDX* for reading files.

`writeWriteX`

Suppresses the use of *SMB_COM_WRITE_ANDX* for writing files.

`writeOpenWrite`

Suppresses the use of *SMB_COM_OPEN_ANDX* batched with *SMB_COM_WRITE_ANDX* for writing files.

`findUnix`

Disables the CIFS Unix extensions for reading directories.

`findTrans2`

Disables the use of *trans2/find* for reading directories.

`fsinfoTrans2`

Suppresses the use of *trans2/query_fs_info* for reading filesystem infos.

`sessionSetup`

Suppresses the session setup command (only used for core dialect)

`treeconAndX`

Suppresses the *TREE_CONNNECT_ANDX* command (*TREE_CONNECT* is used instead).

`setDirDates`

Suppresses setting directory modification dates when files are created or deleted in a directory. This may be useful if the server sets the date automatically when directories are modified.

servers This variable may modify the values configured with *defaultServer* for specific servers. It consists of a dictionary where the keys are the netbios names of servers. The value for each server key is also a dictionary. This dictionary has the same structure as the *defaultServer* dictionary. In addition, the following keys may be used:

ipAddress — This entry may contain an IP address or a DNS name for the server. By default, the netbios name is used for a DNS query. This parameter may be overridden from the *cifsmount* commandline.

netbiosName — This entry is a last chance to change the Netbios name that is sent to the server for a given server.

	tcpPort	You may change the TCP port that is used to connect to the server here. Default is 139, the Netbios session service port.

serverClasses This variable may modify the values configured with *defaultServer* and *servers* after the connection has been established based on the information derived from session setup. The decision can depend on the server's operating system and LAN manager type. The format for this variable is an array of dictionaries. Each dictionary must have all of the following three keys:

OS	This entry contains a matching pattern in shell style syntax (* matches any character sequence,? matches one character, [<characters>] matches any of the given characters and [^<characters>] matches none of the given characters). It is matched against the operating system name derived from session setup.
LanManager	This entry also consists of a matching pattern in shell style syntax. It is matched against the LAN manager name derived from session setup. The operating system name and LAN manager name are printed to *syslog* if log level *info* is enabled.
config	If the previous two patterns match, the content of this variable (which must be a dictionary) is used as a server configuration which may contain all definitions that *defaultServer* may contain. If an option is given, it overrides the respective option from the other configurations. The option *disableSmbs* is an exception: all disabled SMBs add up to give the final list of disabled SMBs.

The array is searched from the first to the last entry. If an entry matches, the corresponding configuration is used and the search is aborted.

6 PAM NTLM

This chapter provides a description of PAM NTLM.

Introduction

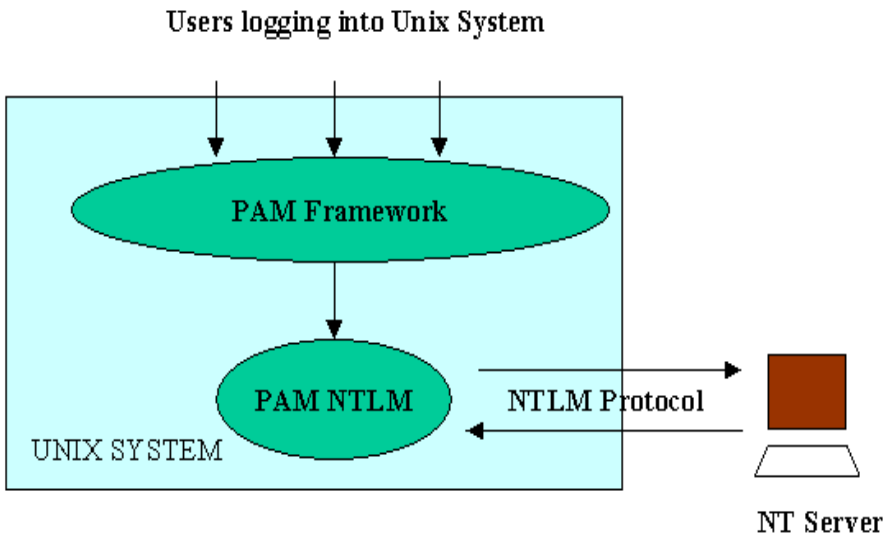
PAM NTLM is a Pluggable Authentication Module (PAM) that enables HP-UX users to be authenticated against Windows servers during system login.

PAM is an authentication framework in UNIX, used to authenticate users logging into a UNIX system. PAM loads a dynamically loadable module (shared library) that performs the actual authentication. PAM can also be configured to use multiple shared library modules.

PAM NTLM uses NT servers to authenticate users logging into an HP-UX system. In other words, PAM NTLM uses the NT LanManager protocol to authenticate the UNIX users. It sends the UNIX user's name and password to the NT server for validation and returns the result to the PAM framework. The CIFS/9000 client users the PAM NTLM authentication information to access the shares on the CIFS/9000 server. Thus users logging into an HP-UX system can access CIFS-mounted file systems without having to use the *cifslogin* command.

Configuring PAM NTLM requires you to understand the PAM framework in general. Refer to *pam(3)*, *pam.conf(4)*, and *Managing Systems and Workgroups* at http://docs.hp.com/hpux/os for more information about PAM.

Figure 6-1 PAM Introduction



PAM NTLM is a dynamically loadable Module. PAM Framework passes user name and password to the PAM NTLM module, which uses the NTLM protocol to authenticate against a Windows server.

PAM NTLM

This section provides a list of PAM NTLM Features and a description of the User Map File.

PAM NTLM Features

- PAM NTLM supports authentication and password management.
- PAM NTLM uses a subset of the Samba *smb.conf* file as its configuration file. See the PAM NTLM Post-installation Instructions below for further information.
- PAM NTLM supports *username mapping* to map a local UNIX user name to a remote NT domain user name to use for authentication. See the PAM NTLM Configuration section for more detailed information.
- Successful user/password authentications are cached for use by the CIFS client.
- Login authentication to CIFS Servers using NTLM encrypted passwords.
- Updating CIFS user passwords on NT 4.0 Primary Domain Controller (PDC) using the HP-UX *passwd(1)* command.

Refer to chapter 2 for installation steps.

User Map File

PAM NTLM supports a *user map file* that maps Unix user names to NT domain user names before authentication by the CIFS server. PAM NTLM will search the user map file for the Unix user name. If found, the mapped NT domain user name will be used to authenticate the user on the CIFS server. The user must enter the correct password for the mapped NT user in order to be authenticated.

If you configure *password(1M)* to use PAM NTLM, then the password of the mapped NT domain user will be changed on the NT server.

PAM NTLM Configuration

Configure the following to set up PAM-NTLM:

- the PAM-NTLM module
- the system file /etc/pam.conf to use the PAM-NTLM module
- a usermap file (*optional*)

Configuring the PAM-NTLM Module

The PAM-NTLM configuration file is */etc/opt/cifsclient/pam/smb.conf*. A default configuration file is also provided (smb.conf.default). Do no change the default configuration file because you may need to refer to it in the future.

Table 6-1

```
##
## Name: smb.conf
##
## Set the values below to the actual names used in your environment
##
## Any line which starts with a semi-colon(;) or a hash(#)
## is a comment and is ignored.
##
##==================== Global Settings ============================
[global]

## workgroup: NT-Domain-Name or Workgroup-Name
   workgroup = workgroup

## password server: the netbios name of the system which will be
## used to authenticate logins.
   password server = pdc_name bdc1_name bdc2_name

## wins server: the system used to locate password servers,
## specified as a fully-qualified DNS name or an IP address.
   wins server = winserv.mycorp.com
```

Configuring the system to use the PAM-NTLM Module

This task consists of editing the global HP-UX PAM configuration file `/etc/pam.conf`.

IMPORTANT

You may not be able to log into the system if PAM is not correctly configured. Make sure that you understand the PAM framework before you modify pam.conf. For information on PAM, see these sections of HP-UX manpages: pam.conf(4), pam_unix(5).

For security reasons, HP strongly recommends you set up your system such that the host system (PAM-UNIX), not the password server configured by PAM-NTLM, authenticates root and other privileged users. HP also recommends using PAM-NTLM services in addition to, not in place of, PAM-UNIX. This configuration is depicted in the sample pam.conf below.

Background Information PAM NTLM provides a centralized authentication service for HP-UX and Microsoft Windows NT servers or other UNIX servers running HP CIFS/9000 server. HP CIFS/9000 Client product includes the PAM NTLM that integrates HP-UX login with any CIFS/9000 server or Windows NT domain controller.

PAM NTLM authenticates the users using encrypted passwords. PAM NTLM also supports password change and password expiry. So users can change their NT password from their HP-UX workstation.

The PAM NTLM consists of two shared PAM NTLM libraries, one provides functionality for all four PAM modules: authentication, account management, session management and password management, and the other, provides support to communicate with NT servers using NT LanManager protocol.

Authentication Module The Authentication Module verifies the identity of a user and sets the user specific credentials. It authenticates users to the NT server (configured in */etc/opt/cifsclient/pam/smb.conf*). If the password matches and the user has rights to login (account is not disabled), he/she is allowed to login to the system.

CIFS/9000 client can use this logon information when the user accesses CIFS mounted shares. So the user doesn't have to use the `cifslogin` command before accessing CIFS shares.

The authentication module supports `use_first_pass, try_first_pass` and debug options. With debug option, PAM NTLM logs debug messages to syslog. Refer to PAM documentation for details on the other options.

Account Management The account management module retrieves the user's password expiration information and verifies that the password has not expired. Except for the above mentioned use, this function does not do any real account management on NT/UX server. This is provided for compatibility with PAM specification.

Session Management The session management module provides functions to initiate and terminate sessions. PAM NTLM does not support Session management and always returns success. It is provided for compatibility with the PAM specification.

Password Management The password management module provides a function to change passwords in the NT server database. The following options may be passed to this PAM module: `use_first_pass, try_first_pass` and debug. With the debug option, PAM NTLM prints debug messages to `syslog`. Refer to the PAM documentation for details on the other options.

PAM-NTLM provides the following services:

- Password Authentication
- Password Change
- Password Change Upon Notice of Expiration

Each service corresponds to a specific section of pam.conf. Add entries for the services you wish to use:

- For Password Authentication, modify the “Authentication management” section of pam.conf.
- For Password Change, modify “Password management.”
- For Password Change Upon Notice of Expiration, modify “Authentication management,” “Password management,” and “Account management” (in order to utilize Password Change Upon Notice of expiration, you must also enable both Password Authentication and Password Change).

The following are sample *pam.conf* files with all three PAM-NTLM services configured. Each PAM-NTLM entry consistes of a line that refers to the shared library *libpam_ntlm.1*. In the `authentication management` section, when PAM-NTLM is used in conjunction with PAM-UNIX, it is recommended that the option `try_first_pass` be specified with the PAM-UNIX entry, as shown.

WARNING **Ensure that you refer to the *pam.conf* file that matches the version of HP-UX installed on your system (use `uname -r` to check the version). In particular, you should add lines to *pam.conf* exactly as shown *without modifying paths*. For versions B.11.22 of HP-UX, paths to the PAM libraries are different than in earlier versions. If incorrect paths are used in *pam.conf*, it can become impossible to login to the system.**

The following sample pam.conf file is for version B.11.22 of HP-UX:

Example 6-1 Sample file for HP-UX version B.11.22

```
=====================================================================
#
# PAM configuration
#
# Authentication management
# Note: For PA applications, /usr/lib/security/libpam_unix.so.1 is a
# symbolic link that points to the corresponding PA PAM module.
#
#
login     auth sufficient   /usr/lib/security/$ISA/libpam_ntlm.so.1
login    auth required     /usr/lib/security/$ISA/libpam_unix.so.1 try_first_pass
su       auth required     /usr/lib/security/$ISA/libpam_unix.so.1
dtlogin  auth required     /usr/lib/security/$ISA/libpam_unix.so.1
dtaction auth required     /usr/lib/security/$ISA/libpam_unix.so.1
ftp      auth required     /usr/lib/security/$ISA/libpam_unix.so.1
OTHER    auth required     /usr/lib/security/$ISA/libpam_unix.so.1
#
# Account management
#
login     auth sufficient        /usr/lib/security/$ISA/libpam_ntlm.so.1
login    account required       /usr/lib/security/$ISA/libpam_unix.so.1
su       account required       /usr/lib/security/$ISA/libpam_unix.so.1
dtlogin  account required       /usr/lib/security/$ISA/libpam_unix.so.1
dtaction account required       /usr/lib/security/$ISA/libpam_unix.so.1
ftp      account required       /usr/lib/security/$ISA/libpam_unix.so.1
#
OTHER    account required       /usr/lib/security/$ISA/libpam_unix.so.1
#
# Session management
#
login    session required       /usr/lib/security/$ISA/libpam_unix.so.1
dtlogin  session required       /usr/lib/security/$ISA/libpam_unix.so.1
dtaction session required       /usr/lib/security/$ISA/libpam_unix.so.1
OTHER    session required       /usr/lib/security/$ISA/libpam_unix.so.1
#
# Password management
#
login    auth sufficient        /usr/lib/security/$ISA/libpam_ntlm.so.1
login    password required      /usr/lib/security/$ISA/libpam_unix.so.1
passwd   password required      /usr/lib/security/$ISA/libpam_unix.so.1
```

```
dtlogin  password required      /usr/lib/security/$ISA/libpam_unix.so.1
dtaction password required      /usr/lib/security/$ISA/libpam_unix.so.1
OTHER    password required      /usr/lib/security/$ISA/libpam_unix.so.1
=====================================================================
```

The following sample pam.conf file is for versions B.11.00 and B.11.11 of HP-UX:

Example 6-2 Sample file for HP-UX versions B.11.00 and B.11.11

```
#
# PAM configuration
#
# Authentication management
#
login      auth sufficient     /usr/lib/security/libpam_ntlm.1
login      auth required       /usr/lib/security/libpam_unix.1 try_first_pass
su         auth required       /usr/lib/security/libpam_unix.1
dtlogin    auth required       /usr/lib/security/libpam_unix.1
dtaction   auth required       /usr/lib/security/libpam_unix.1
ftp        auth required       /usr/lib/security/libpam_unix.1
OTHER      auth required       /usr/lib/security/libpam_unix.1
#
# Account management
#
login      account required    /usr/lib/security/libpam_ntlm.1
login      account required    /usr/lib/security/libpam_unix.1
su         account required    /usr/lib/security/libpam_unix.1
dtlogin    account required    /usr/lib/security/libpam_unix.1
dtaction   account required    /usr/lib/security/libpam_unix.1
ftp        account required    /usr/lib/security/libpam_unix.1
OTHER      account required    /usr/lib/security/libpam_unix.1
#
# Session management
#
login      session required    /usr/lib/security/libpam_unix.1
dtlogin    session required    /usr/lib/security/libpam_unix.1
dtaction   session required    /usr/lib/security/libpam_unix.1
OTHER      session required    /usr/lib/security/libpam_unix.1
#
# Password management
#
login      password sufficient /usr/lib/security/libpam_ntlm.1
login      password required   /usr/lib/security/libpam_unix.1
passwd     password required   /usr/lib/security/libpam_ntlm.1
dtlogin    password required   /usr/lib/security/libpam_unix.1
dtaction   password required   /usr/lib/security/libpam_unix.1
OTHER      password required   /usr/lib/security/libpam_unix.1
```

Configuring a User Map File

To configure PAM NTLM to use the user map file, add the following line to the [Global] section of the */etc/opt/cifsclient/pam/smb.conf* file:

Domain user map = /etc/opt/cifsclient/pam/domain_user.map

You can configure the name and location of the user map file. For name and location, HP recommends the line as shown above.

The format of a domain user file entry is:

UNIXusername = [\\DOMAIN_NAME\\] DomainUserName

UNIXusername is an existing account on the HP-UX system; *DomainUserName* is the name of the user that is mapped in the NT domain. *DOMAIN_NAME* is optional.

The user map file is parsed line by line. If any line begins with a # or *a;* then the line is ignored. Each line should contain a single Unix user name on the left and then a single NT Domain User name on the right, separated by a tabstop or '='. If either name contains spaces then you should enclose it in quotes.

Using NIS Distribution of the User Map File

The user map file is enabled to be distributed via NIS in a similar manner to the distribution of */etc/passwd* to NIS clients.

To use this feature:

1. Convert the master user map file into an NIS map file named domainusermap.byname on the NIS master server.

NOTE

The NIS map file name *domainusermap.byname* is the default name that PAM NTLM uses for the NIS map file. You can configure a different NIS user map name in the PAM NTLM configuration file (*/etc/opt/cifsclient/pam/smb.conf*) of each NIS client. The configuration option is:

nis ntuser mapname = <new usr map filename>

2. In the user map file of each NIS client that will receive the distributed map file, add an entry with the plus sign (+) in the first column of the line. The plus sign is used to indicate that parsing the file should stop at that point and the remaining search of the user map file should use NIS calls to the NIS server.

Glossary

A

ACL Access Control List, meta-data that describes which users are allowed access to file data and what type of access is granted to that data. ACLs define "access rights." In this scheme, users typically belong to "groups," and groups are given access rights as a whole. Typical types of access rights are read (list), write (modify), or create (insert.) Different file systems have varying levels of ACL support and different file systems define different access rights. For example, DOS has only one set of rights for a file (since only one user is considered to use a DOS system). A POSIX 6-compliant file system allows multiple rights to be assigned to multiple files and directories for multiple users and multiple groups of users.

ASP Application service provider, an e-business that essentially "rents" applications to users.

Authentication Scheme to ensure that a user who is accessing file data is indeed the intended user. A secure networked file system uses authentication to prevent access occurring from someone pretending to be the intended user.

Authorization Ensures that a user has access only to file system data that the user has the right to access. Just because a user is authenticated does not mean he or she should be able to read or modify any file. In the simplest form or authorization, users are given read or modify permissions to individual files and directories in a file system, through the use of access control information (called an Access Control List, or ACL.)

C

CIFS Common Internet File System, a specification for a file access protocol designed for the Internet.

CIFS/9000 Hewlett-Packard's implementation of CIFS for UNIX. CIFS/9000 provides both server and client modules for both HP 9000 servers and workstations.

Credential A piece of information that identifies a user. A credential may be as simple as a number that is uniquely associated with a user (like a social security number), or it may be complicated and contain additional identifying information. A strong credential contains proof, sometimes called a verifier, that the user of the credential is indeed the actual user the credential identifies.

D

Diffie-Hellman A protocol used to securely share a secret key between two users. Diffie-Hellman protocol uses a form of public key exchange to share the secret key. Diffie-Hellman is known to be susceptible to an interceptor's attack, but authenticated Diffie-Hellman Key Agreement, a later enhancement, prevents such a middle-person attack.

E

Encryption Encryption ensures that data is viewable only by those who possess a secret (or private) key. Encrypted data is meaningless unless the secret key is used to decrypt the data. Encryption and decryption of data is called ciphering.

I

Integrity Integrity ensures that file system data is not modified by an intruder. An intruder can not intercept a file system data packet and modify it without the network file system discovering and rejecting the tampering.

K

Kerberos An authentication and authorization security system developed by MIT and the IETF working group. It is based on secret key technology, and is generally easier to manage than a public key infrastructure because of its centralized design. However, Kerberos is not as scalable as a public key infrastructure.

P

Public Key An encryption method by which two users exchange data securely, but in one direction only. A user, who has a private key, creates a corresponding public key. This public key can be given to anyone. Anyone who wishes to send encrypted data to the user may encrypt the data using the public key. Only the user who possesses the private key can decrypt the data.

Public Key Infrastructure Method of managing public key encryption. Although public key technology has the advantage of never exchanging decryption keys, it has the disadvantage of being difficult to manage. Some issues include distribution of public keys with proof of the key's ownership, and revocation of expired or terminated keys.

S

Samba An open source product that first appeared in the mid-1990's. Samba provides NT file and print server capability for UNIX systems, including most of the capabilities of Advanced Server for UNIX, with the exception of the Primary Domain Controller (PDC) and Backup Domain Controller (BDC) synchronization protocols. Although Samba is widely used, vendor support for it is not generally available.

Secret Key Secret key, also known as symmetric-key or shared-key, encryption is a ciphering technique by which two users exchange data by encrypting and decrypting data with a shared secret key. Data is both encrypted and decrypted with the same key. The secret key must be exchanged securely (such as through the "cones of silence") since anyone knowing the secret key can decrypt the data.

SMB Server Message Block, the file-sharing protocol at the heart of Windows networking. SMB is shared by Windows NT, Windows 95, Windows for Workgroups, and OS/2 LAN Manager. CIFS is essentially a renaming of this protocol.

U