# SAMBA Project Documentation

**SAMBA Team**

# Abstract

This book is a collection of HOWTOs added to Samba documentation over the years. I try to ensure that all are current, but sometimes the is a larger job than one person can maintain. The most recent version of this document can be found at http://www.samba.org/ on the "Documentation" page. Please send updates to jerry@samba.org.

Cheers, jerry

# Table of Contents

# Table of Contents

# Table of Contents

# Chapter 1. How to Install and Test SAMBA

## 1.1. Step 0: Read the man pages

The man pages distributed with SAMBA contain lots of useful info that will help to get you started. If you don't know how to read man pages then try something like:

```
$ nroff -man smbd.8 | more
```

Other sources of information are pointed to by the Samba web site, http://www.samba.org

---

## 1.2. Step 1: Building the Binaries

To do this, first run the program **./configure** in the source directory. This should automatically configure Samba for your operating system. If you have unusual needs then you may wish to run

```
root# ./configure --help
```

first to see what special options you can enable. Then exectuting

```
root# make
```

will create the binaries. Once it's successfully compiled you can use

```
root# make install
```

to install the binaries and manual pages. You can separately install the binaries and/or man pages using

```
root# make installbin
```

and

```
root# make installman
```

Note that if you are upgrading for a previous version of Samba you might like to know that the old versions of the binaries will be renamed with a ".old" extension. You can go back to the previous version with

```
root# make revert
```

if you find this version a disaster!

---

## 1.3. Step 2: The all important step

At this stage you must fetch yourself a coffee or other drink you find stimulating. Getting the rest of the install right can sometimes be tricky, so you will probably need it.

If you have installed samba before then you can skip this step.

## 1.4. Step 3: Create the smb configuration file.

There are sample configuration files in the examples subdirectory in the distribution. I suggest you read them carefully so you can see how the options go together in practice. See the man page for all the options.

The simplest useful configuration file would be something like this:

```
        [global]
           workgroup = MYGROUP

           [homes]
              guest ok = no
              read only = no
```

which would allow connections by anyone with an account on the server, using either their login name or "homes" as the service name. (Note that I also set the workgroup that Samba is part of. See BROWSING.txt for defails)

Note that **make install** will not install a `smb.conf` file. You need to create it yourself.

Make sure you put the smb.conf file in the same place you specified in the`Makefile` (the default is to look for it in `/usr/local/samba/lib/`).

For more information about security settings for the [homes] share please refer to the document UNIX_SECURITY.txt.

## 1.5. Step 4: Test your config file with testparm

It's important that you test the validity of your `smb.conf` file using the testparm program. If testparm runs OK then it will list the loaded services. If not it will give an error message.

Make sure it runs OK and that the services look resonable before proceeding.

## 1.6. Step 5: Starting the smbd and nmbd

You must choose to start smbd and nmbd either as daemons or from **inetd**. Don't try to do both! Either you can put them in `inetd.conf` and have them started on demand by **inetd**, or you can start them as daemons either from the command line or in `/etc/rc.local`. See the man pages for details on the command line options. Take particular care to read the bit about what user you need to be in order to start Samba. In many cases you must be root.

The main advantage of starting **smbd** and **nmbd** as a daemon is that they will respond slightly more quickly to an initial connection request. This is, however, unlikely to be a problem.

## 1.6.1. Step 5a: Starting from inetd.conf

NOTE; The following will be different if you use NIS or NIS+ to distributed services maps.

Look at your `/etc/services`. What is defined at port 139/tcp. If nothing is defined then add a line like this:

**`netbios-ssn 139/tcp`**

similarly for 137/udp you should have an entry like:

**`netbios-ns 137/udp`**

Next edit your `/etc/inetd.conf` and add two lines something like this:

```
netbios-ssn stream tcp nowait root /usr/local/samba/bin/smbd smbd
netbios-ns dgram udp wait root /usr/local/samba/bin/nmbd nmbd
```

The exact syntax of `/etc/inetd.conf` varies between unixes. Look at the other entries in inetd.conf for a guide.

NOTE: Some unixes already have entries like netbios_ns (note the underscore) in `/etc/services`. You must either edit `/etc/services` or `/etc/inetd.conf` to make them consistant.

NOTE: On many systems you may need to use the "interfaces" option in smb.conf to specify the IP address and netmask of your interfaces. Run **ifconfig** as root if you don't know what the broadcast is for your net. **nmbd** tries to determine it at run time, but fails on somunixes. See the section on "testing nmbd" for a method of finding if you need to do this.

!!!WARNING!!! Many unixes only accept around 5 parameters on the command line in `inetd.conf`. This means you shouldn't use spaces between the options and arguments, or you should use a script, and start the script from **inetd**.

Restart **inetd**, perhaps just send it a HUP. If you have installed an earlier version of **nmbd** then you may need to kill nmbd as well.

---

## 1.6.2. Step 5b. Alternative: starting it as a daemon

To start the server as a daemon you should create a script something like this one, perhaps calling it `startsmb`.

```
#!/bin/sh
/usr/local/samba/bin/smbd -D
/usr/local/samba/bin/nmbd -D
```

then make it executable with **chmod +x startsmb**

You can then run **startsmb** by hand or execute it from `/etc/rc.local`

To kill it send a kill signal to the processes **nmbd** and **smbd**.

NOTE: If you use the SVR4 style init system then you may like to look at the `examples/svr4-startup` script to make Samba fit into that system.

## 1.7. Step 6: Try listing the shares available on your server

```
$ smbclient -L yourhostname
```

Your should get back a list of shares available on your server. If you don't then something is incorrectly setup. Note that this method can also be used to see what shares are available on other LanManager clients (such as WfWg).

If you choose user level security then you may find that Samba requests a password before it will list the shares. See the **smbclient** man page for details. (you can force it to list the shares without a password by adding the option -U% to the command line. This will not work with non-Samba servers)

## 1.8. Step 7: Try connecting with the unix client

```
$ smbclient //yourhostname/aservice
```

Typically the *yourhostname* would be the name of the host where you installed **smbd**. The *aservice* is any service you have defined in the `smb.conf` file. Try your user name if you just have a [homes] section in `smb.conf`.

For example if your unix host is bambi and your login name is fred you would type:

```
$ smbclient //bambi/fred
```

## 1.9. Step 8: Try connecting from a DOS, WfWg, Win9x, WinNT, Win2k, OS/2, etc... client

Try mounting disks. eg:

```
C:\WINDOWS\> net use d: \\servername\service
```

Try printing. eg:

```
C:\WINDOWS\> net use lpt1: \\servername\spoolservice

C:\WINDOWS\> print filename
```

Celebrate, or send me a bug report!

## 1.10. What If Things Don't Work?

If nothing works and you start to think "who wrote this pile of trash" then I suggest you do step 2 again (and again) till you calm down.

Then you might read the file DIAGNOSIS.txt and the FAQ. If you are still stuck then try the mailing list or newsgroup (look in the README for details). Samba has been successfully installed at thousands of sites worldwide, so maybe someone else has hit your problem and has overcome it. You could also use the WWW site to scan back issues of the samba−digest.

When you fix the problem PLEASE send me some updates to the documentation (or source code) so that the next person will find it easier.

---

### 1.10.1. Diagnosing Problems

If you have instalation problems then go to `DIAGNOSIS.txt` to try to find the problem.

---

### 1.10.2. Scope IDs

By default Samba uses a blank scope ID. This means all your windows boxes must also have a blank scope ID. If you really want to use a non−blank scope ID then you will need to use the −i <scope> option to nmbd, smbd, and smbclient. All your PCs will need to have the same setting for this to work. I do not recommend scope IDs.

---

### 1.10.3. Choosing the Protocol Level

The SMB protocol has many dialects. Currently Samba supports 5, called CORE, COREPLUS, LANMAN1, LANMAN2 and NT1.

You can choose what maximum protocol to support in the `smb.conf` file. The default is NT1 and that is the best for the vast majority of sites.

In older versions of Samba you may have found it necessary to use COREPLUS. The limitations that led to this have mostly been fixed. It is now less likely that you will want to use less than LANMAN1. The only remaining advantage of COREPLUS is that for some obscure reason WfWg preserves the case of passwords in this protocol, whereas under LANMAN1, LANMAN2 or NT1 it uppercases all passwords before sending them, forcing you to use the "password level=" option in some cases.

The main advantage of LANMAN2 and NT1 is support for long filenames with some clients (eg: smbclient, Windows NT or Win95).

See the smb.conf(5) manual page for more details.

Note: To support print queue reporting you may find that you have to use TCP/IP as the default protocol under WfWg. For some reason if you leave Netbeui as the default it may break the print queue reporting on some systems. It is presumably a WfWg bug.

## 1.10.4. Printing from UNIX to a Client PC

To use a printer that is available via a smb−based server from a unix host you will need to compile the smbclient program. You then need to install the script "smbprint". Read the instruction in smbprint for more details.

There is also a SYSV style script that does much the same thing called smbprint.sysv. It contains instructions.

## 1.10.5. Locking

One area which sometimes causes trouble is locking.

There are two types of locking which need to be performed by a SMB server. The first is "record locking" which allows a client to lock a range of bytes in a open file. The second is the "deny modes" that are specified when a file is open.

Samba supports "record locking" using the fcntl() unix system call. This is often implemented using rpc calls to a rpc.lockd process running on the system that owns the filesystem. Unfortunately many rpc.lockd implementations are very buggy, particularly when made to talk to versions from other vendors. It is not uncommon for the rpc.lockd to crash.

There is also a problem translating the 32 bit lock requests generated by PC clients to 31 bit requests supported by most unixes. Unfortunately many PC applications (typically OLE2 applications) use byte ranges with the top bit set as semaphore sets. Samba attempts translation to support these types of applications, and the translation has proved to be quite successful.

Strictly a SMB server should check for locks before every read and write call on a file. Unfortunately with the way fcntl() works this can be slow and may overstress the rpc.lockd. It is also almost always unnecessary as clients are supposed to independently make locking calls before reads and writes anyway if locking is important to them. By default Samba only makes locking calls when explicitly asked to by a client, but if you set "strict locking = yes" then it will make lock checking calls on every read and write.

You can also disable by range locking completely using "locking = no". This is useful for those shares that don't support locking or don't need it (such as cdroms). In this case Samba fakes the return codes of locking calls to tell clients that everything is OK.

The second class of locking is the "deny modes". These are set by an application when it opens a file to determine what types of access should be allowed simultaneously with its open. A client may ask for DENY_NONE, DENY_READ, DENY_WRITE or DENY_ALL. There are also special compatability modes called DENY_FCB and DENY_DOS.

You can disable share modes using "share modes = no". This may be useful on a heavily loaded server as the share modes code is very slow. See also the FAST_SHARE_MODES option in the Makefile for a way to do full share modes very fast using shared memory (if your OS supports it).

## 1.10.6. Mapping Usernames

If you have different usernames on the PCs and the unix server then take a look at the "username map" option. See the smb.conf man page for details.

## 1.10.7. Other Character Sets

If you have problems using filenames with accented characters in them (like the German, French or Scandinavian character sets) then I recommmend you look at the "valid chars" option in smb.conf and also take a look at the validchars package in the examples directory.

# Chapter 2. LanMan and NT Password Encryption in Samba 2.x

## 2.1. Introduction

With the development of LanManager and Windows NT compatible password encryption for Samba, it is now able to validate user connections in exactly the same way as a LanManager or Windows NT server.

This document describes how the SMB password encryption algorithm works and what issues there are in choosing whether you want to use it. You should read it carefully, especially the part about security and the "PROS and CONS" section.

---

## 2.2. How does it work?

LanManager encryption is somewhat similar to UNIX password encryption. The server uses a file containing a hashed value of a user's password. This is created by taking the user's plaintext password, capitalising it, and either truncating to 14 bytes or padding to 14 bytes with null bytes. This 14 byte value is used as two 56 bit DES keys to encrypt a 'magic' eight byte value, forming a 16 byte value which is stored by the server and client. Let this value be known as the "hashed password".

Windows NT encryption is a higher quality mechanism, consisting of doing an MD4 hash on a Unicode version of the user's password. This also produces a 16 byte hash value that is non−reversible.

When a client (LanManager, Windows for WorkGroups, Windows 95 or Windows NT) wishes to mount a Samba drive (or use a Samba resource), it first requests a connection and negotiates the protocol that the client and server will use. In the reply to this request the Samba server generates and appends an 8 byte, random value − this is stored in the Samba server after the reply is sent and is known as the "challenge". The challenge is different for every client connection.

The client then uses the hashed password (16 byte values described above), appended with 5 null bytes, as three 56 bit DES keys, each of which is used to encrypt the challenge 8 byte value, forming a 24 byte value known as the "response".

In the SMB call SMBsessionsetupX (when user level security is selected) or the call SMBtconX (when share level security is selected), the 24 byte response is returned by the client to the Samba server. For Windows NT protocol levels the above calculation is done on both hashes of the user's password and both responses are returned in the SMB call, giving two 24 byte values.

The Samba server then reproduces the above calculation, using its own stored value of the 16 byte hashed password (read from the `smbpasswd` file − described later) and the challenge value that it kept from the negotiate protocol reply. It then checks to see if the 24 byte value it calculates matches the 24 byte value returned to it from the client.

If these values match exactly, then the client knew the correct password (or the 16 byte hashed value − see security note below) and is thus allowed access. If not, then the client did not know the correct password and is denied access.

Note that the Samba server never knows or stores the cleartext of the user's password − just the 16 byte hashed values derived from it. Also note that the cleartext password or 16 byte hashed values are never transmitted over the network − thus increasing security.

## 2.3. Important Notes About Security

The unix and SMB password encryption techniques seem similar on the surface. This similarity is, however, only skin deep. The unix scheme typically sends clear text passwords over the nextwork when logging in. This is bad. The SMB encryption scheme never sends the cleartext password over the network but it does store the 16 byte hashed values on disk. This is also bad. Why? Because the 16 byte hashed values are a "password equivalent". You cannot derive the user's password from them, but they could potentially be used in a modified client to gain access to a server. This would require considerable technical knowledge on behalf of the attacker but is perfectly possible. You should thus treat the smbpasswd file as though it contained the cleartext passwords of all your users. Its contents must be kept secret, and the file should be protected accordingly.

Ideally we would like a password scheme which neither requires plain text passwords on the net or on disk. Unfortunately this is not available as Samba is stuck with being compatible with other SMB systems (WinNT, WfWg, Win95 etc).

**Warning**

Note that Windows NT 4.0 Service pack 3 changed the default for permissible authentication so that plaintext passwords are *never* sent over the wire. The solution to this is either to switch to encrypted passwords with Samba or edit the Windows NT registry to re−enable plaintext passwords. See the document WinNT.txt for details on how to do this.

Other Microsoft operating systems which also exhibit this behavior includes

- MS DOS Network client 3.0 with the basic network redirector installed
- Windows 95 with the network redirector update installed
- Windows 98 [se]
- Windows 2000

*Note :*All current release of Microsoft SMB/CIFS clients support authentication via the SMB Challenge/Response mechanism described here. Enabling clear text authentication does not disable the ability of the client to particpate in encrypted authentication.

### 2.3.1. Advantages of SMB Encryption

- plain text passwords are not passed across the network. Someone using a network sniffer cannot just record passwords going to the SMB server.
- WinNT doesn't like talking to a server that isn't using SMB encrypted passwords. It will refuse to browse the server if the server is also in user level security mode. It will insist on prompting the user for the password on each connection, which is very annoying. The only things you can do to stop this is to use SMB encryption.

### 2.3.2. Advantages of non−encrypted passwords

- plain text passwords are not kept on disk.
- uses same password file as other unix services such as login and ftp
- you are probably already using other services (such as telnet and ftp) which send plain text passwords over the net, so sending them for SMB isn't such a big deal.

---

## 2.4. The smbpasswd file

In order for Samba to participate in the above protocol it must be able to look up the 16 byte hashed values given a user name. Unfortunately, as the UNIX password value is also a one way hash function (ie. it is impossible to retrieve the cleartext of the user's password given the UNIX hash of it), a separate password file containing this 16 byte value must be kept. To minimise problems with these two password files, getting out of sync, the UNIX `/etc/passwd` and the `smbpasswd` file, a utility, **mksmbpasswd.sh**, is provided to generate a smbpasswd file from a UNIX `/etc/passwd` file.

To generate the smbpasswd file from your `/etc/passwd` file use the following command :

```
$ cat /etc/passwd | mksmbpasswd.sh  > /usr/local/samba/private/smbpasswd
```

If you are running on a system that uses NIS, use

```
$ ypcat passwd | mksmbpasswd.sh  > /usr/local/samba/private/smbpasswd
```

The **mksmbpasswd.sh** program is found in the Samba source directory. By default, the smbpasswd file is stored in :

```
/usr/local/samba/private/smbpasswd
```

The owner of the `/usr/local/samba/private/` directory should be set to root, and the permissions on it should be set to 0500 (**chmod 500 /usr/local/samba/private**).

Likewise, the smbpasswd file inside the private directory should be owned by root and the permissions on is should be set to 0600 (**chmod 600 smbpasswd**).

The format of the smbpasswd file is (The line has been wrapped here. It should appear as one entry per line in your smbpasswd file.)

```
username:uid:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX:
        [Account type]:LCT−<last−change−time>:Long name
```

Although only the *username*, *uid*, *XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*, [*Account type*] and *last−change−time* sections are significant and are looked at in the Samba code.

It is *VITALLY* important that there by 32 'X' characters between the two ':' characters in the XXX sections − the smbpasswd and Samba code will fail to validate any entries that do not have 32 characters between ':' characters. The first XXX section is for the Lanman password hash, the second is for the Windows NT version.

When the password file is created all users have password entries consisting of 32 'X' characters. By default this disallows any access as this user. When a user has a password set, the 'X' characters change to 32 ascii hexadecimal digits (0−9, A−F). These are an ascii representation of the 16 byte hashed value of a user's password.

To set a user to have no password (not recommended), edit the file using vi, and replace the first 11 characters with the ascii text `"NO PASSWORD"` (minus the quotes).

For example, to clear the password for user bob, his smbpasswd file entry would look like :

```
bob:100:NO PASSWORDXXXXXXXXXXXXXXXXXXXXX:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX:[U          ]:L
```

If you are allowing users to use the smbpasswd command to set their own passwords, you may want to give users NO PASSWORD initially so they do not have to enter a previous password when changing to their new password (not recommended). In order for you to allow this the **smbpasswd** program must be able to connect to the **smbd** daemon as that user with no password. Enable this by adding the line :

**null passwords = yes**

to the [global] section of the smb.conf file (this is why the above scenario is not recommended). Preferably, allocate your users a default password to begin with, so you do not have to enable this on your server.

*Note :* This file should be protected very carefully. Anyone with access to this file can (with enough knowledge of the protocols) gain access to your SMB server. The file is thus more sensitive than a normal unix `/etc/passwd` file.

---

## 2.5. The smbpasswd Command

The smbpasswd command maintains the two 32 byte password fields in the smbpasswd file. If you wish to make it similar to the unix **passwd** or **yppasswd** programs, install it in `/usr/local/samba/bin/` (or your main Samba binary directory).

Note that as of Samba 1.9.18p4 this program *MUST NOT BE INSTALLED* setuid root (the new **smbpasswd** code enforces this restriction so it cannot be run this way by accident).

**smbpasswd** now works in a client−server mode where it contacts the local smbd to change the user's password on its behalf. This has enormous benefits − as follows.

- smbpasswd no longer has to be setuid root − an enormous range of potential security problems is eliminated.
- **smbpasswd** now has the capability to change passwords on Windows NT servers (this only works when the request is sent to the NT Primary Domain Controller if you are changing an NT Domain user's password).

To run smbpasswd as a normal user just type :

```
$ smbpasswd
```

`Old SMB password:` **`<type old value here - or hit return if there was no old password>`**

`New SMB Password:` **`<type new value>`**

`Repeat New SMB Password:` **`<re-type new value`**

If the old value does not match the current value stored for that user, or the two new values do not match each other, then the password will not be changed.

If invoked by an ordinary user it will only allow the user to change his or her own Samba password.

If run by the root user smbpasswd may take an optional argument, specifying the user name whose SMB password you wish to change. Note that when run as root smbpasswd does not prompt for or check the old password value, thus allowing root to set passwords for users who have forgotten their passwords.

**smbpasswd** is designed to work in the same way and be familiar to UNIX users who use the **passwd** or **yppasswd** commands.

For more details on using **smbpasswd** refer to the man page which will always be the definitive reference.

---

## 2.6. Setting up Samba to support LanManager Encryption

This is a very brief description on how to setup samba to support password encryption.

1. compile and install samba as usual
2. enable encrypted passwords in `smb.conf` by adding the line **encrypt passwords = yes** in the [global] section
3. create the initial `smbpasswd` password file in the place you specified in the Makefile (--prefix=<dir>). See the notes under the The smbpasswd File section earlier in the document for details.

Note that you can test things using smbclient.

---

# Chapter 3. Hosting a Microsoft Distributed File System tree on Samba

## 3.1. Instructions

The Distributed File System (or Dfs) provides a means of separating the logical view of files and directories that users see from the actual physical locations of these resources on the network. It allows for higher availability, smoother storage expansion, load balancing etc. For more information about Dfs, refer to Microsoft documentation.

This document explains how to host a Dfs tree on a Unix machine (for Dfs−aware clients to browse) using Samba.

To enable SMB−based DFS for Samba, configure it with the *−−with−msdfs* option. Once built, a Samba server can be made a Dfs server by setting the global boolean *host msdfs* parameter in the `smb.conf` file. You designate a share as a Dfs root using the share level boolean *msdfs root* parameter. A Dfs root directory on Samba hosts Dfs links in the form of symbolic links that point to other servers. For example, a symbolic link `junction−>msdfs:storage1\share1` in the share directory acts as the Dfs junction. When Dfs−aware clients attempt to access the junction link, they are redirected to the storage location (in this case, \\storage1\share1).

Dfs trees on Samba work with all Dfs−aware clients ranging from Windows 95 to 2000.

Here's an example of setting up a Dfs tree on a Samba server.

```
# The smb.conf file:
[global]
        netbios name = SAMBA
        host msdfs   = yes

[dfs]
        path = /export/dfsroot
        msdfs root = yes
```

In the /export/dfsroot directory we set up our dfs links to other servers on the network.

```
root# cd /export/dfsroot

root# chown root /export/dfsroot

root# chmod 755 /export/dfsroot

root# ln −s msdfs:storageA\\shareA linka

root# ln −s msdfs:serverB\\share,serverC\\share linkb
```

You should set up the permissions and ownership of the directory acting as the Dfs root such that only designated users can create, delete or modify the msdfs links. Also note that symlink names should be all lowercase. This limitation exists to have Samba avoid trying all the case combinations to get at the link name.

Finally set up the symbolic links to point to the network shares you want, and start Samba.

Users on Dfs−aware clients can now browse the Dfs tree on the Samba server at \\samba\dfs. Accessing links linka or linkb (which appear as directories to the client) takes users directly to the appropriate shares on the network.

---

## 3.1.1. Notes

- Windows clients need to be rebooted if a previously mounted non−dfs share is made a dfs root or vice versa. A better way is to introduce a new share and make it the dfs root.
- Currently there's a restriction that msdfs symlink names should all be lowercase.
- For security purposes, the directory acting as the root of the Dfs tree should have ownership and permissions set so that only designated users can modify the symbolic links in the directory.

---

# Chapter 4. Printing Support in Samba 2.2.x

## 4.1. Introduction

Beginning with the 2.2.0 release, Samba supports the native Windows NT printing mechanisms implemented via MS−RPC (i.e. the SPOOLSS named pipe). Previous versions of Samba only supported LanMan printing calls.

The additional functionality provided by the new SPOOLSS support includes:

- Support for downloading printer driver files to Windows 95/98/NT/2000 clients upon demand.
- Uploading of printer drivers via the Windows NT Add Printer Wizard (APW) or the Imprints tool set (refer to http://imprints.sourceforge.net).
- Support for the native MS−RPC printing calls such as StartDocPrinter, EnumJobs(), etc... (See the MSDN documentation at http://msdn.microsoft.com/ for more information on the Win32 printing API)
- Support for NT Access Control Lists (ACL) on printer objects
- Improved support for printer queue manipulation through the use of an internal databases for spooled job information

There has been some initial confusion about what all this means and whether or not it is a requirement for printer drivers to be installed on a Samba host in order to support printing from Windows clients. A bug existed in Samba 2.2.0 which made Windows NT/2000 clients require that the Samba server possess a valid driver for the printer. This is fixed in Samba 2.2.1 and once again, Windows NT/2000 clients can use the local APW for installing drivers to be used with a Samba served printer. This is the same behavior exhibited by Windows 9x clients. As a side note, Samba does not use these drivers in any way to process spooled files. They are utilized entirely by the clients.

The following MS KB article, may be of some help if you are dealing with Windows 2000 clients: *How to Add Printers with No User Interaction in Windows 2000*

http://support.microsoft.com/support/kb/articles/Q189/1/05.ASP

---

## 4.2. Configuration

| **[print$] vs. [printer$]** |
| --- |
| Previous versions of Samba recommended using a share named [printer$]. This name was taken from the printer$ service created by Windows 9x clients when a printer was shared. Windows 9x printer servers always have a printer$ service which provides read−only access via no password in order to support printer driver downloads.<br><br>However, the initial implementation allowed for a parameter named `printer driver location` to be used on a per share basis to specify the location of the driver files associated with that printer. Another parameter named `printer driver` provided a means of defining the printer driver name to be sent to the client. |

These parameters, including *printer driver file* parameter, are being depreciated and should not be used in new installations. For more information on this change, you should refer to the Migration section of this document.

## 4.2.1. Creating [print$]

In order to support the uploading of printer driver files, you must first configure a file share named [print$]. The name of this share is hard coded in Samba's internals so the name is very important (print$ is the service used by Windows NT print servers to provide support for printer driver download).

You should modify the server's smb.conf file to create the following file share (of course, some of the parameter values, such as 'path' are arbitrary and should be replaced with appropriate values for your site):

```
[print$]
    path = /usr/local/samba/printers
    guest ok = yes
    browseable = yes
    read only = yes
    ; since this share is configured as read only, then we need
    ; a 'write list'.  Check the file system permissions to make
    ; sure this account can copy files to the share.  If this
    ; is setup to a non-root account, then it should also exist
    ; as a 'printer admin'
    write list = ntadmin
```

The *write list* is used to allow administrative level user accounts to have write access in order to update files on the share. See the smb.conf(5) man page for more information on configuring file shares.

The requirement for **guest ok = yes** depends upon how your site is configured. If users will be guaranteed to have an account on the Samba host, then this is a non−issue.

> **Author's Note:** The non−issue is that if all your Windows NT users are guaranteed to be authenticated by the Samba server (such as a domain member server and the NT user has already been validated by the Domain Controller in order to logon to the Windows NT console), then guest access is not necessary. Of course, in a workgroup environment where you just want to be able to print without worrying about silly accounts and security, then configure the share for guest access. You'll probably want to add **map to guest = Bad User** in the [global] section as well. Make sure you understand what this parameter does before using it though. −−jerry

In order for a Windows NT print server to support the downloading of driver files by multiple client architectures, it must create subdirectories within the [print$] service which correspond to each of the supported client architectures. Samba follows this model as well.

Next create the directory tree below the [print$] share for each architecture you wish to support.

```
[print$]-----
        |-W32X86           ; "Windows NT x86"
        |-WIN40            ; "Windows 95/98"
        |-W32ALPHA         ; "Windows NT Alpha_AXP"
        |-W32MIPS          ; "Windows NT R4000"
        |-W32PPC           ; "Windows NT PowerPC"
```

| ATTENTION! REQUIRED PERMISSIONS |
| --- |
| In order to currently add a new driver to you Samba host, one of two conditions must hold true:<br>• The account used to connect to the Samba host must have a uid of 0 (i.e. a root account)<br>• The account used to connect to the Samba host must be a member of the *printer admin* list.<br>Of course, the connected account must still possess access to add files to the subdirectories beneath [print$]. Remember that all file shares are set to 'read only' by default. |

Once you have created the required [print$] service and associated subdirectories, simply log onto the Samba server using a root (or *printer admin*) account from a Windows NT 4.0 client. Navigate to the "Printers" folder on the Samba server. You should see an initial listing of printers that matches the printer shares defined on your Samba host.

## 4.2.2. Setting Drivers for Existing Printers

The initial listing of printers in the Samba host's Printers folder will have no real printer driver assigned to them. By default, in Samba 2.2.0 this driver name was set to *NO PRINTER DRIVER AVAILABLE FOR THIS PRINTER*. Later versions changed this to a NULL string to allow the use tof the local Add Printer Wizard on NT/2000 clients. Attempting to view the printer properties for a printer which has this default driver assigned will result in the error message:

*Device settings cannot be displayed. The driver for the specified printer is not installed, only spooler properties will be displayed. Do you want to install the driver now?*

Click "No" in the error dialog and you will be presented with the printer properties window. The way assign a driver to a printer is to either

- Use the "New Driver..." button to install a new printer driver, or
- Select a driver from the popup list of installed drivers. Initially this list will be empty.

If you wish to install printer drivers for client operating systems other than "Windows NT x86", you will need to use the "Sharing" tab of the printer properties dialog.

Assuming you have connected with a root account, you will also be able modify other printer properties such as ACLs and device settings using this dialog box.

A few closing comments for this section, it is possible on a Windows NT print server to have printers listed in the Printers folder which are not shared. Samba does not make this distinction. By definition, the only printers of which Samba is aware are those which are specified as shares in `smb.conf`.

Another interesting side note is that Windows NT clients do not use the SMB printer share, but rather can print directly to any printer on another Windows NT host using MS−RPC. This of course assumes that the printing client has the necessary privileges on the remote host serving the printer. The default permissions assigned by Windows NT to a printer gives the "Print" permissions to the "Everyone" well−known group.

### 4.2.3. Support a large number of printers

One issue that has arisen during the development phase of Samba 2.2 is the need to support driver downloads for 100's of printers. Using the Windows NT APW is somewhat awkward to say the list. If more than one printer are using the same driver, the **rpcclient's setdriver command** can be used to set the driver associated with an installed driver. The following is example of how this could be accomplished:

```
$ rpcclient pogo -U root%secret -c "enumdrivers"
Domain=[NARNIA] OS=[Unix] Server=[Samba 2.2.0-alpha3]

[Windows NT x86]
Printer Driver Info 1:
     Driver Name: [HP LaserJet 4000 Series PS]

Printer Driver Info 1:
     Driver Name: [HP LaserJet 2100 Series PS]

Printer Driver Info 1:
     Driver Name: [HP LaserJet 4Si/4SiMX PS]

$ rpcclient pogo -U root%secret -c "enumprinters"
Domain=[NARNIA] OS=[Unix] Server=[Samba 2.2.0-alpha3]
     flags:[0x800000]
     name:[\\POGO\hp-print]
     description:[POGO\\POGO\hp-print,NO DRIVER AVAILABLE FOR THIS PRINTER,]
     comment:[]

$ rpcclient pogo -U root%secret \
>  -c "setdriver hp-print \"HP LaserJet 4000 Series PS\""
Domain=[NARNIA] OS=[Unix] Server=[Samba 2.2.0-alpha3]
Successfully set hp-print to driver HP LaserJet 4000 Series PS.
```

---

### 4.2.4. Adding New Printers via the Windows NT APW

By default, Samba offers all printer shares defined in `smb.conf` in the "Printers..." folder. Also existing in this folder is the Windows NT Add Printer Wizard icon. The APW will be show only if

- The connected user is able to successfully execute an OpenPrinterEx(\\server) with administrative priviledges (i.e. root or *`printer admin`*).
- *`show add printer wizard = yes`* (the default).

In order to be able to use the APW to successfully add a printer to a Samba server, the *`add printer command`* must have a defined value. The program hook must successfully add the printer to the system (i.e. `/etc/printcap` or appropriate files) and `smb.conf` if necessary.

When using the APW from a client, if the named printer share does not exist, **smbd** will execute the *`add printer command`* and reparse to the `smb.conf` to attempt to locate the new printer share. If the share is still not defined, an error of "Access Denied" is returned to the client. Note that the *`add printer program`* is executed under the context of the connected user, not necessarily a root account.

There is a complementing *`delete printer command`* for removing entries from the "Printers..." folder.

---

### 4.2.5. Samba and Printer Ports

Windows NT/2000 print servers associate a port with each printer. These normally take the form of LPT1:, COM1:, FILE:, etc... Samba must also support the concept of ports associated with a printer. By default, only one printer port, named "Samba Printer Port", exists on a system. Samba does not really a port in order to print, rather it is a requirement of Windows clients.

Note that Samba does not support the concept of "Printer Pooling" internally either. This is when a logical printer is assigned to multiple ports as a form of load balancing or fail over.

If you require that multiple ports be defined for some reason, `smb.conf` possesses a *`enumports command`* which can be used to define an external program that generates a listing of ports on a system.

---

## 4.3. The Imprints Toolset

The Imprints tool set provides a UNIX equivalent of the Windows NT Add Printer Wizard. For complete information, please refer to the Imprints web site at http://imprints.sourceforge.net/ as well as the documentation included with the imprints source distribution. This section will only provide a brief introduction to the features of Imprints.

---

### 4.3.1. What is Imprints?

Imprints is a collection of tools for supporting the goals of

- Providing a central repository information regarding Windows NT and 95/98 printer driver packages
- Providing the tools necessary for creating the Imprints printer driver packages.
- Providing an installation client which will obtain and install printer drivers on remote Samba and Windows NT 4 print servers.

---

### 4.3.2. Creating Printer Driver Packages

The process of creating printer driver packages is beyond the scope of this document (refer to Imprints.txt also included with the Samba distribution for more information). In short, an Imprints driver package is a gzipped tarball containing the driver files, related INF files, and a control file needed by the installation client.

---

### 4.3.3. The Imprints server

The Imprints server is really a database server that may be queried via standard HTTP mechanisms. Each printer entry in the database has an associated URL for the actual downloading of the package. Each package is digitally signed via GnuPG which can be used to verify that package downloaded is actually the one referred in the Imprints database. It is *not* recommended that this security check be disabled.

---

### 4.3.4. The Installation Client

More information regarding the Imprints installation client is available in the `Imprints-Client-HOWTO.ps` file included with the imprints source package.

The Imprints installation client comes in two forms.

- a set of command line Perl scripts
- a GTK+ based graphical interface to the command line perl scripts

The installation client (in both forms) provides a means of querying the Imprints database server for a matching list of known printer model names as well as a means to download and install the drivers on remote Samba and Windows NT print servers.

The basic installation process is in four steps and perl code is wrapped around **smbclient** and **rpcclient**.

```
foreach (supported architecture for a given driver)
{
     1.  rpcclient: Get the appropriate upload directory
         on the remote server
     2.  smbclient: Upload the driver files
     3.  rpcclient: Issues an AddPrinterDriver() MS-RPC
}

4.  rpcclient: Issue an AddPrinterEx() MS-RPC to actually
    create the printer
```

One of the problems encountered when implementing the Imprints tool set was the name space issues between various supported client architectures. For example, Windows NT includes a driver named "Apple LaserWriter II NTX v51.8" and Windows 95 callsits version of this driver "Apple LaserWriter II NTX"

The problem is how to know what client drivers have been uploaded for a printer. As astute reader will remember that the Windows NT Printer Properties dialog only includes space for one printer driver name. A quick look in the Windows NT 4.0 system registry at

`HKLM\System\CurrentControlSet\Control\Print\Environment`

will reveal that Windows NT always uses the NT driver name. The is ok as Windows NT always requires that at least the Windows NT version of the printer driver is present. However, Samba does not have the requirement internally. Therefore, how can you use the NT driver name if is has not already been installed?

The way of sidestepping this limitation is to require that all Imprints printer driver packages include both the Intel Windows NT and 95/98 printer drivers and that NT driver is installed first.

---

## 4.4. Migration to from Samba 2.0.x to 2.2.x

Given that printer driver management has changed (we hope improved) in 2.2 over prior releases, migration from an existing setup to 2.2 can follow several paths.

Windows clients have a tendency to remember things for quite a while. For example, if a Windows NT client has attached to a Samba 2.0 server, it will remember the server as a LanMan printer server. Upgrading the

Samba host to 2.2 makes support for MSRPC printing possible, but the NT client will still remember the previous setting.

In order to give an NT client printing "amesia" (only necessary if you want to use the newer MSRPC printing functionality in Samba), delete the registry keys associated with the print server contained in `[HKLM\SYSTEM\CurrentControlSet\Control\Print]`. The spooler service on the client should be stopped prior to doing this:

```
C:\WINNT\ > net stop spooler
```

*All the normal disclaimers about editing the registry go here.* Be careful, and know what you are doing.

The spooler service should be restarted after you have finished removing the appropriate registry entries by replacing the **stop** command above with **start**.

Windows 9x clients will continue to use LanMan printing calls with a 2.2 Samba server so there is no need to perform any of these modifications on non−NT clients.

| **Achtung!** |
| --- |
| The following smb.conf parameters are considered to be depreciated and will be removed soon. Do not use them in new installations<br>• *printer driver file (G)*<br>• *printer driver (S)*<br>• *printer driver location (S)* |

Here are the possible scenarios for supporting migration:

- If you do not desire the new Windows NT print driver support, nothing needs to be done. All existing parameters work the same.
- If you want to take advantage of NT printer driver support but do not want to migrate the 9x drivers to the new setup, the leave the existing printers.def file. When smbd attempts to locate a 9x driver for the printer in the TDB and fails it will drop down to using the printers.def (and all associated parameters). The **make_printerdef** tool will also remain for backwards compatibility but will be moved to the "this tool is the old way of doing it" pile.
- If you install a Windows 9x driver for a printer on your Samba host (in the printing TDB), this information will take precedence and the three old printing parameters will be ignored (including print driver location).
- If you want to migrate an existing `printers.def` file into the new setup, the current only solution is to use the Windows NT APW to install the NT drivers and the 9x drivers. This can be scripted using **smbclient** and **rpcclient**. See the Imprints installation client at http://imprints.sourceforge.net/ for an example.

# Chapter 5. security = domain in Samba 2.x

## 5.1. Joining an NT Domain with Samba 2.2

In order for a Samba−2 server to join an NT domain, you must first add the NetBIOS name of the Samba server to the NT domain on the PDC using Server Manager for Domains. This creates the machine account in the domain (PDC) SAM. Note that you should add the Samba server as a "Windows NT Workstation or Server", *NOT* as a Primary or backup domain controller.

Assume you have a Samba−2 server with a NetBIOS name of `SERV1` and are joining an NT domain called `DOM`, which has a PDC with a NetBIOS name of `DOMPDC` and two backup domain controllers with NetBIOS names `DOMBDC1` and `DOMBDC2` .

In order to join the domain, first stop all Samba daemons and run the command:

```
root# smbpasswd −j DOM −r DOMPDC
```

as we are joining the domain DOM and the PDC for that domain (the only machine that has write access to the domain SAM database) is DOMPDC. If this is successful you will see the message:

```
smbpasswd: Joined domain DOM.
```

in your terminal window. See the smbpasswd(8) man page for more details.

There is existing development code to join a domain without having to create the machine trust account on the PDC beforehand. This code will hopefully be available soon in release branches as well.

This command goes through the machine account password change protocol, then writes the new (random) machine account password for this Samba server into a file in the same directory in which an smbpasswd file would be stored − normally :

```
/usr/local/samba/private
```

In Samba 2.0.x, the filename looks like this:

```
<NT DOMAIN NAME>.<Samba Server Name>.mac
```

The `.mac` suffix stands for machine account password file. So in our example above, the file would be called:

```
DOM.SERV1.mac
```

In Samba 2.2, this file has been replaced with a TDB (Trivial Database) file named `secrets.tdb`.

This file is created and owned by root and is not readable by any other user. It is the key to the domain−level security for your system, and should be treated as carefully as a shadow password file.

Now, before restarting the Samba daemons you must edit your `smb.conf(5)` file to tell Samba it should now use domain security.

Change (or add) your *`security =`* line in the [global] section of your smb.conf to read:

**security = domain**

Next change the *workgroup =* line in the [global] section to read:

**workgroup = DOM**

as this is the name of the domain we are joining.

You must also have the parameter *encrypt passwords* set to `yes` in order for your users to authenticate to the NT PDC.

Finally, add (or modify) a *password server =* line in the [global] section to read:

**password server = DOMPDC DOMBDC1 DOMBDC2**

These are the primary and backup domain controllers Samba will attempt to contact in order to authenticate users. Samba will try to contact each of these servers in order, so you may want to rearrange this list in order to spread out the authentication load among domain controllers.

Alternatively, if you want smbd to automatically determine the list of Domain controllers to use for authentication, you may set this line to be :

**password server = ***

This method, which was introduced in Samba 2.0.6, allows Samba to use exactly the same mechanism that NT does. This method either broadcasts or uses a WINS database in order to find domain controllers to authenticate against.

Finally, restart your Samba daemons and get ready for clients to begin using domain security!

---

## 5.2. Samba and Windows 2000 Domains

Many people have asked regarding the state of Samba's ability to participate in a Windows 2000 Domain. Samba 2.2 is able to act as a member server of a Windows 2000 domain operating in mixed or native mode.

There is much confusion between the circumstances that require a "mixed" mode Win2k DC and a when this host can be switched to "native" mode. A "mixed" mode Win2k domain controller is only needed if Windows NT BDCs must exist in the same domain. By default, a Win2k DC in "native" mode will still support NetBIOS and NTLMv1 for authentication of legacy clients such as Windows 9x and NT 4.0. Samba has the same requirements as a Windows NT 4.0 member server.

The steps for adding a Samba 2.2 host to a Win2k domain are the same as those for adding a Samba server to a Windows NT 4.0 domain. The only exception is that the "Server Manager" from NT 4 has been replaced by the "Active Directory Users and Computers" MMC (Microsoft Management Console) plugin.

---

## 5.3. Why is this better than security = server?

Currently, domain security in Samba doesn't free you from having to create local Unix users to represent the users attaching to your server. This means that if domain user `DOM\fred` attaches to your domain security Samba server, there needs to be a local Unix user fred to represent that user in the Unix filesystem. This is very similar to the older Samba security mode security = server, where Samba would pass through the authentication request to a Windows NT server in the same way as a Windows 95 or Windows 98 server would.

Please refer to the Winbind paper for information on a system to automatically assign UNIX uids and gids to Windows NT Domain users and groups. This code is available in development branches only at the moment, but will be moved to release branches soon.

The advantage to domain−level security is that the authentication in domain−level security is passed down the authenticated RPC channel in exactly the same way that an NT server would do it. This means Samba servers now participate in domain trust relationships in exactly the same way NT servers do (i.e., you can add Samba servers into a resource domain and have the authentication passed on from a resource domain PDC to an account domain PDC.

In addition, with **security = server** every Samba daemon on a server has to keep a connection open to the authenticating server for as long as that daemon lasts. This can drain the connection resources on a Microsoft NT server and cause it to run out of available connections. With **security = domain**, however, the Samba daemons connect to the PDC/BDC only for as long as is necessary to authenticate the user, and then drop the connection, thus conserving PDC connection resources.

And finally, acting in the same manner as an NT server authenticating to a PDC means that as part of the authentication reply, the Samba server gets the user identification information such as the user SID, the list of NT groups the user belongs to, etc. All this information will allow Samba to be extended in the future into a mode the developers currently call appliance mode. In this mode, no local Unix users will be necessary, and Samba will generate Unix uids and gids from the information passed back from the PDC when a user is authenticated, making a Samba server truly plug and play in an NT domain environment. Watch for this code soon.

*NOTE:* Much of the text of this document was first published in the Web magazine LinuxWorld as the article Doing the NIS/NT Samba.

---

# Chapter 6. How to Configure Samba 2.2 as a Primary Domain Controller

## 6.1. Background

> **Note:** *Author's Note :* This document is a combination of David Bannon's Samba 2.2 PDC HOWTO and the Samba NT Domain FAQ. Both documents are superceeded by this one.

Version of Samba prior to release 2.2 had marginal capabilities to act as a Windows NT 4.0 Primary Domain Controller (PDC). The following functionality should work in 2.2:

- domain logons for Windows NT 4.0/2000 clients
- placing a Windows 9x client in user level security
- retrieving a list of users and groups from a Samba PDC to Windows 9x/NT/2000 clients
- roving (roaming) user profiles
- Windows NT 4.0 style system policies

The following pieces of functionality are not included in the 2.2 release:

- Windows NT 4 domain trusts
- SAM replication with Windows NT 4.0 Domain Controllers (i.e. a Samba PDC and a Windows NT BDC or vice versa)
- Adding users via the User Manager for Domains
- Acting as a Windows 2000 Domain Controller (i.e. Kerberos and Active Directory)

Please note that Windows 9x clients are not true members of a domain for reasons outlined in this article. Therefore the protocol for support Windows 9x style domain logons is completely different from NT4 domain logons and has been officially supported for some time.

Beginning with Samba 2.2.0, we are proud to announce official support for Windows NT 4.0 style domain logons from Windows NT 4.0 and Windows 2000 (including SP1) clients. This article outlines the steps necessary for configuring Samba as a PDC. It is necessary to have a working Samba server prior to implementing the PDC functionality. If you have not followed the steps outlined in UNIX_INSTALL.html, please make sure that your server is configured correctly before proceeding. Another good resource in the smb.conf(5) man page.

Implementing a Samba PDC can basically be divided into 2 broad steps.

1. Configuring the Samba PDC
2. Creating machine trust accounts and joining clients to the domain

There are other minor details such as user profiles, system policies, etc... However, these are not necessarily specific to a Samba PDC as much as they are related to Windows NT networking concepts. They will be mentioned only briefly here.

---

## 6.2. Configuring the Samba Domain Controller

The first step in creating a working Samba PDC is to understand the parameters necessary in smb.conf. I will not attempt to re−explain the parameters here as they are more that adequately covered in the smb.conf man page. For convenience, the parameters have been linked with the actual smb.conf description.

Here is an example smb.conf for acting as a PDC:

```
[global]
    ; Basic server settings
    netbios name = POGO
    workgroup = NARNIA

    ; we should act as the domain and local master browser
    os level = 64
    preferred master = yes
    domain master = yes
    local master = yes

    ; security settings (must user security = user)
    security = user

    ; encrypted passwords are a requirement for a PDC
    encrypt passwords = yes

    ; support domain logons
    domain logons = yes

    ; where to store user profiles?
    logon path = \\%N\profiles\%u

    ; where is a user's home directory and where should it
    ; be mounted at?
    logon drive = H:
    logon home = \\homeserver\%u

    ; specify a generic logon script for all users
    ; this is a relative **DOS** path to the [netlogon] share
    logon script = logon.cmd

; necessary share for domain controller
[netlogon]
    path = /usr/local/samba/lib/netlogon
    writeable = no
    write list = ntadmin

; share for storing user profiles
[profiles]
    path = /export/smb/ntprofile
    writeable = yes
    create mask = 0600
    directory mask = 0700
```

There are a couple of points to emphasize in the above configuration.

- Encrypted passwords must be enabled. For more details on how to do this, refer to ENCRYPTION.html.
- The server must support domain logons and a `[netlogon]` share

- The server must be the domain master browser in order for Windows client to locate the server as a DC.

As Samba 2.2 does not offer a complete implementation of group mapping between Windows NT groups and UNIX groups (this is really quite complicated to explain in a short space), you should refer to the domain admin users and domain admin group smb.conf parameters for information of creating a Domain Admins style accounts.

---

## 6.3. Creating Machine Trust Accounts and Joining Clients to the Domain

A machine trust account is a user account owned by a computer. The account password acts as the shared secret for secure communication with the Domain Controller. Hence the reason that a Windows 9x host is never a true member of a domain because it does not posses a machine trust account and thus has no shared secret with the DC.

On a Windows NT PDC, these machine trust account passwords are stored in the registry. A Samba PDC stores these accounts in he same location as user LanMan and NT password hashes (currently `smbpasswd`). However, machine trust accounts only possess and use the NT password hash.

There are two means of creating machine trust accounts.

- Manual creation before joining the client to the domain. In this case, the password is set to a known value -- the lower case of the machine's netbios name.
- Creation of the account at the time of joining the domain. In this case, the session key of the administrative account used to join the client to the domain acts as an encryption key for setting the password to a random value.

Because Samba requires machine accounts to possess a UNIX uid from which an Windows NT SID can be generated, all of these accounts will have an entry in `/etc/passwd` and smbpasswd. Future releases will alleviate the need to create `/etc/passwd` entries.

The `/etc/passwd` entry will list the machine name with a $ appended, won't have a passwd, will have a null shell and no home directory. For example a machine called 'doppy' would have an `/etc/passwd` entry like this :

```
doppy$:x:505:501:NTMachine:/dev/null:/bin/false
```

If you are manually creating the machine accounts, it is necessary to add the `/etc/passwd` (or NIS passwd map) entry prior to adding the `smbpasswd` entry. The following command will create a new machine account ready for use.

`root#` smbpasswd -a -m *`machine_name`*

where *`machine_name`* is the machine's netbios name.

*If you manually create a machine account, immediately join the client to the domain.* An open account like this can allow intruders to gain access to user account information in your domain.

The second way of creating machine trust accounts is to add them on the fly at the time the client is joined to

the domain. You will need to include a value for the add user script parameter. Below is an example I use on a RedHat 6.2 Linux system.

```
add user script = /usr/sbin/useradd -d /dev/null -g 100 -s /bin/false -M %u
```

In Samba 2.2, *only the root account* can be used to create machine accounts on the fly like this. Therefore, it is required to create an entry in smbpasswd for *root*. The password *SHOULD* be set to s different password that the associated `/etc/passwd` entry for security reasons.

---

## 6.4. Common Problems and Errors

*I cannot include a '$' in a machine name.*

A 'machine name' in (typically) `/etc/passwd` of the machine name with a '$' appended. FreeBSD (and other BSD systems ?) won't create a user with a '$' in their name.

The problem is only in the program used to make the entry, once made, it works perfectly. So create a user without the '$' and use **vipw** to edit the entry, adding the '$'. Or create the whole entry with vipw if you like, make sure you use a unique uid !

*I get told "You already have a connection to the Domain...." or "Cannot join domain, the credentials supplied conflict with an existing set.." when creating a machine account.*

This happens if you try to create a machine account from the machine itself and already have a connection (e.g. mapped drive) to a share (or IPC$) on the Samba PDC. The following command will remove all network drive connections:

`C:\WINNT\>` **net use * /d**

Further, if the machine is a already a 'member of a workgroup' that is the same name as the domain you are joining (bad idea) you will get this message. Change the workgroup name to something else, it does not matter what, reboot, and try again.

*"The system can not log you on (C000019B)...."*

I joined the domain successfully but after upgrading to a newer version of the Samba code I get the message, "The system can not log you on (C000019B), Please try a gain or consult your system administrator" when attempting to logon.

This occurs when the domain SID stored in `private/WORKGROUP.SID` is changed. For example, you remove the file and **smbd** automatically creates a new one. Or you are swapping back and forth between versions 2.0.7, TNG and the HEAD branch code (not recommended). The only way to correct the problem is to restore the original domain SID or remove the domain client from the domain and rejoin.

*"The machine account for this computer either does not exist or is not accessible."*

When I try to join the domain I get the message "The machine account for this computer either does not exist or is not accessible". Whats wrong ?

This problem is caused by the PDC not having a suitable machine account. If you are using the *add user script* method to create accounts then this would indicate that it has not worked. Ensure the domain admin user system is working.

Alternatively if you are creating account entries manually then they have not been created correctly. Make sure that you have the entry correct for the machine account in smbpasswd file on the Samba PDC. If you added the account using an editor rather than using the smbpasswd utility, make sure that the account name is the machine netbios name with a '$' appended to it ( ie. computer_name$ ). There must be an entry in both /etc/passwd and the smbpasswd file. Some people have reported that inconsistent subnet masks between the Samba server and the NT client have caused this problem. Make sure that these are consistent for both client and server.

*When I attempt to login to a Samba Domain from a NT4/W2K workstation, I get a message about my account being disabled.*

This problem is caused by a PAM related bug in Samba 2.2.0. This bug is fixed in 2.2.1. Other symptoms could be unaccessible shares on NT/W2K member servers in the domain or the following error in your smbd.log: passdb/pampass.c:pam_account(268) PAM: UNKNOWN ERROR for User: %user%

At first be ensure to enable the useraccounts with **smbpasswd -e %user%**, this is normaly done, when you create an account.

In order to work around this problem in 2.2.0, configure the *account* control flag in `/etc/pam.d/samba` file as follows:

```
account required        pam_permit.so
```

If you want to remain backward compatibility to samba 2.0.x use `pam_permit.so`, it's also possible to use `pam_pwdb.so`. There are some bugs if you try to use `pam_unix.so`, if you need this, be ensure to use the most recent version of this file.

---

## 6.5. System Policies and Profiles

Much of the information necessary to implement System Policies and Roving User Profiles in a Samba domain is the same as that for implementing these same items in a Windows NT 4.0 domain. You should read the white paper Implementing Profiles and Policies in Windows NT 4.0 available from Microsoft.

Here are some additional details:

*What about Windows NT Policy Editor ?*

To create or edit `ntconfig.pol` you must use the NT Server Policy Editor, **poledit.exe** which is included with NT Server but *not NT Workstation*. There is a Policy Editor on a NTws but it is not suitable for creating *Domain Policies*. Further, although the Windows 95 Policy Editor can be installed on an NT Workstation/Server, it will not work with NT policies because the registry key that are set by the policy templates. However, the files from the NT Server will run happily enough on an NTws. You need `poledit.exe, common.adm` and `winnt.adm`. It is convenient to put the two *.adm files in `c:\winnt\inf` which is where the binary will look for them unless told otherwise. Note also that that directory is 'hidden'.

The Windows NT policy editor is also included with the Service Pack 3 (and later) for Windows NT 4.0. Extract the files using **servicepackname /x**, ie thats **Nt4sp6ai.exe /x** for service pack 6a. The policy editor, **poledit.exe** and the associated template files (*.adm) should be extracted as well. It is also possible to downloaded the policy template files for Office97 and get a copy of the policy editor. Another possible location is with the Zero Administration Kit available for download from Microsoft.

*Can Win95 do Policies ?*

Install the group policy handler for Win9x to pick up group policies. Look on the Win98 CD in `\tools\reskit\netadmin\poledit`. Install group policies on a Win9x client by double−clicking `grouppol.inf`. Log off and on again a couple of times and see if Win98 picks up group policies. Unfortunately this needs to be done on every Win9x machine that uses group policies....

If group policies don't work one reports suggests getting the updated (read: working) grouppol.dll for Windows 9x. The group list is grabbed from /etc/group.

*How do I get 'User Manager' and 'Server Manager'*

Since I don't need to buy an NT Server CD now, how do I get the 'User Manager for Domains', the 'Server Manager' ?

Microsoft distributes a version of these tools called nexus for installation on Windows 95 systems. The tools set includes

- Server Manager
- User Manager for Domains
- Event Viewer

Click here to download the archived file ftp://ftp.microsoft.com/Softlib/MSLFILES/NEXUS.EXE

The Windows NT 4.0 version of the 'User Manager for Domains' and 'Server Manager' are available from Microsoft via ftp from ftp://ftp.microsoft.com/Softlib/MSLFILES/SRVTOOLS.EXE

---

## 6.6. What other help can I get ?

There are many sources of information available in the form of mailing lists, RFC's and documentation. The docs that come with the samba distribution contain very good explanations of general SMB topics such as browsing.

*What are some diagnostics tools I can use to debug the domain logon process and where can I find them?*

One of the best diagnostic tools for debugging problems is Samba itself. You can use the −d option for both smbd and nmbd to specifiy what 'debug level' at which to run. See the man pages on smbd, nmbd and smb.conf for more information on debugging options. The debug level can range from 1 (the default) to 10 (100 for debugging passwords).

Another helpful method of debugging is to compile samba using the **gcc −g** flag. This will include debug information in the binaries and allow you to attach gdb to the running smbd / nmbd process. In order to attach gdb to an smbd process for an NT workstation, first get the workstation to make the connection. Pressing

ctrl−alt−delete and going down to the domain box is sufficient (at least, on the first time you join the domain) to generate a 'LsaEnumTrustedDomains'. Thereafter, the workstation maintains an open connection, and therefore there will be an smbd process running (assuming that you haven't set a really short smbd idle timeout) So, in between pressing ctrl alt delete, and actually typing in your password, you can gdb attach and continue.

Some useful samba commands worth investigating:

- testparam | more
- smbclient −L //{netbios name of server}

An SMB enabled version of tcpdump is available from http://www.tcpdup.org/. Ethereal, another good packet sniffer for UNIX and Win32 hosts, can be downloaded from http://www.ethereal.com.

For tracing things on the Microsoft Windows NT, Network Monitor (aka. netmon) is available on the Microsoft Developer Network CD's, the Windows NT Server install CD and the SMS CD's. The version of netmon that ships with SMS allows for dumping packets between any two computers (ie. placing the network interface in promiscuous mode). The version on the NT Server install CD will only allow monitoring of network traffic directed to the local NT box and broadcasts on the local subnet. Be aware that Ethereal can read and write netmon formatted files.

*How do I install 'Network Monitor' on an NT Workstation or a Windows 9x box?*

Installing netmon on an NT workstation requires a couple of steps. The following are for installing Netmon V4.00.349, which comes with Microsoft Windows NT Server 4.0, on Microsoft Windows NT Workstation 4.0. The process should be similar for other version of Windows NT / Netmon. You will need both the Microsoft Windows NT Server 4.0 Install CD and the Workstation 4.0 Install CD.

Initially you will need to install 'Network Monitor Tools and Agent' on the NT Server. To do this

- Goto Start − Settings − Control Panel − Network − Services − Add
- Select the 'Network Monitor Tools and Agent' and click on 'OK'.
- Click 'OK' on the Network Control Panel.
- Insert the Windows NT Server 4.0 install CD when prompted.

At this point the Netmon files should exist in `%SYSTEMROOT%\System32\netmon\*.*`. Two subdirectories exist as well, `parsers\` which contains the necessary DLL's for parsing the netmon packet dump, and `captures\`.

In order to install the Netmon tools on an NT Workstation, you will first need to install the 'Network Monitor Agent' from the Workstation install CD.

- Goto Start − Settings − Control Panel − Network − Services − Add
- Select the 'Network Monitor Agent' and click on 'OK'.
- Click 'OK' on the Network Control Panel.
- Insert the Windows NT Workstation 4.0 install CD when prompted.

Now copy the files from the NT Server in %SYSTEMROOT%\System32\netmon*.* to %SYSTEMROOT%\System32\netmon*.* on the Workstation and set permissions as you deem appropriate for your site. You will need administrative rights on the NT box to run netmon.

To install Netmon on a Windows 9x box install the network monitor agent from the Windows 9x CD (\admin\nettools\netmon). There is a readme file located with the netmon driver files on the CD if you need information on how to do this. Copy the files from a working Netmon installation.

---

## 6.6.1. URLs and similar

- Home of Samba site http://samba.org. We have a mirror near you !
- The *Development* document on the Samba mirrors might mention your problem. If so, it might mean that the developers are working on it.
- See how Scott Merrill simulates a BDC behavior at http://www.skippy.net/linux/smb−howto.html.
- Although 2.0.7 has almost had its day as a PDC, David Bannon will keep the 2.0.7 PDC pages at http://bioserve.latrobe.edu.au/samba going for a while yet.
- Misc links to CIFS information http://samba.org/cifs/
- NT Domains for Unix http://mailhost.cb1.com/~lkcl/ntdom/
- FTP site for older SMB specs: ftp://ftp.microsoft.com/developr/drg/CIFS/

---

## 6.6.2. Mailing Lists

*How do I get help from the mailing lists ?*

There are a number of Samba related mailing lists. Go to http://samba.org, click on your nearest mirror and then click on **Support** and then click on **Samba related mailing lists**.

For questions relating to Samba TNG go to http://www.samba−tng.org/ It has been requested that you don't post questions about Samba−TNG to the main stream Samba lists.

If you post a message to one of the lists please observe the following guide lines :

- Always remember that the developers are volunteers, they are not paid and they never guarantee to produce a particular feature at a particular time. Any time lines are 'best guess' and nothing more.
- Always mention what version of samba you are using and what operating system its running under. You should probably list the relevant sections of your smb.conf file, at least the options in [global] that affect PDC support.
- In addition to the version, if you obtained Samba via CVS mention the date when you last checked it out.
- Try and make your question clear and brief, lots of long, convoluted questions get deleted before they are completely read ! Don't post html encoded messages (if you can select colour or font size its html).
- If you run one of those nifty 'I'm on holidays' things when you are away, make sure its configured to not answer mailing lists.
- Don't cross post. Work out which is the best list to post to and see what happens, ie don't post to both samba−ntdom and samba−technical. Many people active on the lists subscribe to more than one list and get annoyed to see the same message two or more times. Often someone will see a message and thinking it would be better dealt with on another, will forward it on for you.
- You might include *partial* log files written at a debug level set to as much as 20. Please don't send the entire log but enough to give the context of the error messages.
- (Possibly) If you have a complete netmon trace ( from the opening of the pipe to the error ) you can send the *.CAP file as well.

- Please think carefully before attaching a document to an email. Consider pasting the relevant parts into the body of the message. The samba mailing lists go to a huge number of people, do they all need a copy of your smb.conf in their attach directory ?

*How do I get off the mailing lists ?*

To have your name removed from a samba mailing list, go to the same place you went to to get on it. Go to http://lists.samba.org, click on your nearest mirror and then click on **Support** and then click on **Samba related mailing lists**. Or perhaps see here

Please don't post messages to the list asking to be removed, you will just be referred to the above address (unless that process failed in some way...)

---

## 6.7. DOMAIN_CONTROL.txt : Windows NT Domain Control & Samba

This appendix was originally authored by John H Terpstra of the Samba Team and is included here for posterity.

*NOTE :* The term "Domain Controller" and those related to it refer to one specific method of authentication that can underly an SMB domain. Domain Controllers prior to Windows NT Server 3.1 were sold by various companies and based on private extensions to the LAN Manager 2.1 protocol. Windows NT introduced Microsoft−specific ways of distributing the user authentication database. See DOMAIN.txt for examples of how Samba can participate in or create SMB domains based on shared authentication database schemes other than the Windows NT SAM.

Windows NT Server can be installed as either a plain file and print server (WORKGROUP workstation or server) or as a server that participates in Domain Control (DOMAIN member, Primary Domain controller or Backup Domain controller).

The same is true for OS/2 Warp Server, Digital Pathworks and other similar products, all of which can participate in Domain Control along with Windows NT. However only those servers which have licensed Windows NT code in them can be a primary Domain Controller (eg Windows NT Server, Advanced Server for Unix.)

To many people these terms can be confusing, so let's try to clear the air.

Every Windows NT system (workstation or server) has a registry database. The registry contains entries that describe the initialization information for all services (the equivalent of Unix Daemons) that run within the Windows NT environment. The registry also contains entries that tell application software where to find dynamically loadable libraries that they depend upon. In fact, the registry contains entries that describes everything that anything may need to know to interact with the rest of the system.

The registry files can be located on any Windows NT machine by opening a command prompt and typing:

```
C:\WINNT\> dir %SystemRoot%\System32\config
```

The environment variable %SystemRoot% value can be obtained by typing:

```
C:\WINNT>echo %SystemRoot%
```

The active parts of the registry that you may want to be familiar with are the files called: default, system, software, sam and security.

In a domain environment, Microsoft Windows NT domain controllers participate in replication of the SAM and SECURITY files so that all controllers within the domain have an exactly identical copy of each.

The Microsoft Windows NT system is structured within a security model that says that all applications and services must authenticate themselves before they can obtain permission from the security manager to do what they set out to do.

The Windows NT User database also resides within the registry. This part of the registry contains the user's security identifier, home directory, group memberships, desktop profile, and so on.

Every Windows NT system (workstation as well as server) will have its own registry. Windows NT Servers that participate in Domain Security control have a database that they share in common − thus they do NOT own an independent full registry database of their own, as do Workstations and plain Servers.

The User database is called the SAM (Security Access Manager) database and is used for all user authentication as well as for authentication of inter− process authentication (ie: to ensure that the service action a user has requested is permitted within the limits of that user's privileges).

The Samba team have produced a utility that can dump the Windows NT SAM into smbpasswd format: see ENCRYPTION.txt for information on smbpasswd and /pub/samba/pwdump on your nearest Samba mirror for the utility. This facility is useful but cannot be easily used to implement SAM replication to Samba systems.

Windows for Workgroups, Windows 95, and Windows NT Workstations and Servers can participate in a Domain security system that is controlled by Windows NT servers that have been correctly configured. At most every domain will have ONE Primary Domain Controller (PDC). It is desirable that each domain will have at least one Backup Domain Controller (BDC).

The PDC and BDCs then participate in replication of the SAM database so that each Domain Controlling participant will have an up to date SAM component within its registry.

---

# Chapter 7. Unifed Logons between Windows NT and UNIX using Winbind

## 7.1. Abstract

Integration of UNIX and Microsoft Windows NT through a unified logon has been considered a "holy grail" in heterogeneous computing environments for a long time. We present *winbind* , a component of the Samba suite of programs as a solution to the unied logon problem. Winbind uses a UNIX implementation of Microsoft RPC calls, Pluggable Authentication Modules, and the Name Service Switch to allow Windows NT domain users to appear and operate as UNIX users on a UNIX machine. This paper describes the winbind system, explaining the functionality it provides, how it is configured, and how it works internally.

---

## 7.2. Introduction

It is well known that UNIX and Microsoft Windows NT have different models for representing user and group information and use different technologies for implementing them. This fact has made it difficult to integrate the two systems in a satisfactory manner.

One common solution in use today has been to create identically named user accounts on both the UNIX and Windows systems and use the Samba suite of programs to provide file and print services between the two. This solution is far from perfect however, as adding and deleting users on both sets of machines becomes a chore and two sets of passwords are required both of which which can lead to synchronization problems between the UNIX and Windows systems and confusion for users.

We divide the unifed logon problem for UNIX machines into three smaller problems:

- Obtaining Windows NT user and group information
- Authenticating Windows NT users
- Password changing for Windows NT users

Ideally, a prospective solution to the unified logon problem would satisfy all the above components without duplication of information on the UNIX machines and without creating additional tasks for the system administrator when maintaining users and groups on either system. The winbind system provides a simple and elegant solution to all three components of the unifed logon problem.

---

## 7.3. What Winbind Provides

Winbind unifies UNIX and Windows NT account management by allowing a UNIX box to become a full member of a NT domain. Once this is done the UNIX box will see NT users and groups as if they were native UNIX users and groups, allowing the NT domain to be used in much the same manner that NIS+ is used within UNIX−only environments.

The end result is that whenever any program on the UNIX machine asks the operating system to lookup a user or group name, the query will be resolved by asking the NT domain controller for the specied domain to do the lookup. Because Winbind hooks into the operating system at a low level (via the NSS name resolution modules in the C library) this redirection to the NT domain controller is completely transparent.

Users on the UNIX machine can then use NT user and group names as they would use "native" UNIX names. They can chown files so that they are owned by NT domain users or even login to the UNIX machine and run a UNIX X−Window session as a domain user.

The only obvious indication that Winbind is being used is that user and group names take the form DOMAIN\user and DOMAIN\group. This is necessary as it allows Winbind to determine that redirection to a domain controller is wanted for a particular lookup and which trusted domain is being referenced.

Additionally, Winbind provides a authentication service that hooks into the Pluggable Authentication Modules (PAM) system to provide authentication via a NT domain to any PAM enabled applications. This capability solves the problem of synchronizing passwords between systems as all passwords are stored in a single location (on the domain controller).

---

## 7.3.1. Target Uses

Winbind is targeted at organizations that have an existing NT based domain infrastructure into which they wish to put UNIX workstations or servers. Winbind will allow these organizations to deploy UNIX workstations without having to maintain a separate account infrastructure. This greatly simplies the administrative overhead of deploying UNIX workstations into a NT based organization.

Another interesting way in which we expect Winbind to be used is as a central part of UNIX based appliances. Appliances that provide file and print services to Microsoft based networks will be able to use Winbind to provide seamless integration of the appliance into the domain.

---

## 7.4. How Winbind Works

The winbind system is designed around a client/server architecture. A long running **winbindd** daemon listens on a UNIX domain socket waiting for requests to arrive. These requests are generated by the NSS and PAM clients and processed sequentially.

The technologies used to implement winbind are described in detail below.

---

## 7.4.1. Microsoft Remote Procedure Calls

Over the last two years, efforts have been underway by various Samba Team members to decode various aspects of the Microsoft Remote Procedure Call (MSRPC) system. This system is used for most network related operations between Windows NT machines including remote management, user authentication and print spooling. Although initially this work was done to aid the implementation of Primary Domain Controller (PDC) functionality in Samba, it has also yielded a body of code which can be used for other purposes.

Winbind uses various MSRPC calls to enumerate domain users and groups and to obtain detailed information about individual users or groups. Other MSRPC calls can be used to authenticate NT domain users and to change user passwords. By directly querying a Windows PDC for user and group information, winbind maps the NT account information onto UNIX user and group names.

---

## 7.4.2. Name Service Switch

The Name Service Switch, or NSS, is a feature that is present in many UNIX operating systems. It allows system information such as hostnames, mail aliases and user information to be resolved from dierent sources. For example, a standalone UNIX workstation may resolve system information from a series of flat files stored on the local lesystem. A networked workstation may first attempt to resolve system information from local files, then consult a NIS database for user information or a DNS server for hostname information.

The NSS application programming interface allows winbind to present itself as a source of system information when resolving UNIX usernames and groups. Winbind uses this interface, and information obtained from a Windows NT server using MSRPC calls to provide a new source of account enumeration. Using standard UNIX library calls, one can enumerate the users and groups on a UNIX machine running winbind and see all users and groups in a NT domain plus any trusted domain as though they were local users and groups.

The primary control le for NSS is `/etc/nsswitch.conf` . When a UNIX application makes a request to do a lookup the C library looks in `/etc/nsswitch.conf` for a line which matches the service type being requested, for example the "passwd" service type is used when user or group names are looked up. This config line species which implementations of that service should be tried andin what order. If the passwd config line is:

**passwd: files example**

then the C library will first load a module called `/lib/libnss_files.so` followed by the module `/lib/libnss_example.so`. The C library will dynamically load each of these modules in turn and call resolver functions within the modules to try to resolve the request. Once the request is resolved the C library returns the result to the application.

This NSS interface provides a very easy way for Winbind to hook into the operating system. All that needs to be done is to put `libnss_winbind.so` in `/lib/` then add "winbind" into `/etc/nsswitch.conf` at the appropriate place. The C library will then call Winbind to resolve user and group names.

---

## 7.4.3. Pluggable Authentication Modules

Pluggable Authentication Modules, also known as PAM, is a system for abstracting authentication and authorization technologies. With a PAM module it is possible to specify different authentication methods for dierent system applications without having to recompile these applications. PAM is also useful for implementing a particular policy for authorization. For example, a system administrator may only allow console logins from users stored in the local password file but only allow users resolved from a NIS database to log in over the network.

Winbind uses the authentication management and password management PAM interface to integrate Windows NT users into a UNIX system. This allows Windows NT users to log in to a UNIX machine and be authenticated against a suitable Primary Domain Controller. These users can also change their passwords and have this change take eect directly on the Primary Domain Controller.

PAM is congured by providing control files in the directory `/etc/pam.d/` for each of the services that require authentication. When an authentication request is made by an application the PAM code in the C library looks up this control file to determine what modules to load to do the authentication check and in what

order. This interface makes adding a new authentication service for Winbind very easy, all that needs to be done is that the `pam_winbind.so` module is copied to `/lib/security/` and the pam control files for relevant services are updated to allow authentication via winbind. See the PAM documentation for more details.

---

### 7.4.4. User and Group ID Allocation

When a user or group is created under Windows NT is it allocated a numerical relative identier (RID). This is slightly dierent to UNIX which has a range of numbers which are used to identify users, and the same range in which to identify groups. It is winbind's job to convert RIDs to UNIX id numbers and vice versa. When winbind is congured it is given part of the UNIX user id space and a part of the UNIX group id space in which to store Windows NT users and groups. If a Windows NT user is resolved for the first time, it is allocated the next UNIX id from the range. The same process applies for Windows NT groups. Over time, winbind will have mapped all Windows NT users and groups to UNIX user ids and group ids.

The results of this mapping are stored persistently in a ID mapping database held in a tdb database). This ensures that RIDs are mapped to UNIX IDs in a consistent way.

---

### 7.4.5. Result Caching

An active system can generate a lot of user and group name lookups. To reduce the network cost of these lookups winbind uses a caching scheme based on the SAM sequence number supplied by NT domain controllers. User or group information returned by a PDC is cached by winbind along with a sequence number also returned by the PDC. This sequence number is incremented by Windows NT whenever any user or group information is modied. If a cached entry has expired, the sequence number is requested from the PDC and compared against the sequence number of the cached entry. If the sequence numbers do not match, then the cached information is discarded and up to date information is requested directly from the PDC.

---

## 7.5. Installation and Configuration

The easiest way to install winbind is by using the packages provided in the `pub/samba/appliance/` directory on your nearest Samba mirror. These packages provide snapshots of the Samba source code and binaries already setup to provide the full functionality of winbind. This setup is a little more complex than a normal Samba build as winbind needs a small amount of functionality from a development code branch called SAMBA_TNG.

Once you have installed the packages you should read the **winbindd(8)** man page which will provide you with conguration information and give you sample conguration files. You may also wish to update the main Samba daemons smbd and nmbd) with a more recent development release, such as the recently announced Samba 2.2 alpha release.

---

## 7.6. Limitations

Winbind has a number of limitations in its current released version which we hope to overcome in future releases:

- Winbind is currently only available for the Linux operating system, although ports to other operating systems are certainly possible. For such ports to be feasible, we require the C library of the target operating system to support the Name Service Switch and Pluggable Authentication Modules systems. This is becoming more common as NSS and PAM gain support among UNIX vendors.
- The mappings of Windows NT RIDs to UNIX ids is not made algorithmically and depends on the order in which unmapped users or groups are seen by winbind. It may be difficult to recover the mappings of rid to UNIX id mapping if the file containing this information is corrupted or destroyed.
- Currently the winbind PAM module does not take into account possible workstation and logon time restrictions that may be been set for Windows NT users.
- Building winbind from source is currently quite tedious as it requires combining source code from two Samba branches. Work is underway to solve this by providing all the necessary functionality in the main Samba code branch.

## 7.7. Conclusion

The winbind system, through the use of the Name Service Switch, Pluggable Authentication Modules, and appropriate Microsoft RPC calls have allowed us to provide seamless integration of Microsoft Windows NT domain users on a UNIX system. The result is a great reduction in the administrative cost of running a mixed UNIX and NT network.

# Chapter 8. UNIX Permission Bits and WIndows NT Access Control Lists

## 8.1. Viewing and changing UNIX permissions using the NT security dialogs

New in the Samba 2.0.4 release is the ability for Windows NT clients to use their native security settings dialog box to view and modify the underlying UNIX permissions.

Note that this ability is careful not to compromise the security of the UNIX host Samba is running on, and still obeys all the file permission rules that a Samba administrator can set.

In Samba 2.0.4 and above the default value of the parameter *nt acl support* has been changed from `false` to `true`, so manipulation of permissions is turned on by default.

---

## 8.2. How to view file security on a Samba share

From an NT 4.0 client, single−click with the right mouse button on any file or directory in a Samba mounted drive letter or UNC path. When the menu pops−up, click on the *Properties* entry at the bottom of the menu. This brings up the normal file properties dialog box, but with Samba 2.0.4 this will have a new tab along the top marked *Security*. Click on this tab and you will see three buttons, *Permissions*, *Auditing*, and *Ownership*. The *Auditing* button will cause either an error message A requested privilege is not held by the client to appear if the user is not the NT Administrator, or a dialog which is intended to allow an Administrator to add auditing requirements to a file if the user is logged on as the NT Administrator. This dialog is non−functional with a Samba share at this time, as the only useful button, the **Add** button will not currently allow a list of users to be seen.

---

## 8.3. Viewing file ownership

Clicking on the **"Ownership"** button brings up a dialog box telling you who owns the given file. The owner name will be of the form :

**"SERVER\user (Long name)"**

Where *SERVER* is the NetBIOS name of the Samba server, *user* is the user name of the UNIX user who owns the file, and *(Long name)* is the discriptive string identifying the user (normally found in the GECOS field of the UNIX password database). Click on the **Close** button to remove this dialog.

If the parameter *nt acl support* is set to `false` then the file owner will be shown as the NT user **"Everyone"**.

The **Take Ownership** button will not allow you to change the ownership of this file to yourself (clicking on it will display a dialog box complaining that the user you are currently logged onto the NT client cannot be found). The reason for this is that changing the ownership of a file is a privilaged operation in UNIX, available only to the *root* user. As clicking on this button causes NT to attempt to change the ownership of a file to the current user logged into the NT client this will not work with Samba at this time.

There is an NT chown command that will work with Samba and allow a user with Administrator privillage connected to a Samba 2.0.4 server as root to change the ownership of files on both a local NTFS filesystem or remote mounted NTFS or Samba drive. This is available as part of the *Seclib* NT security library written by Jeremy Allison of the Samba Team, available from the main Samba ftp site.

## 8.4. Viewing file or directory permissions

The third button is the **"Permissions"** button. Clicking on this brings up a dialog box that shows both the permissions and the UNIX owner of the file or directory. The owner is displayed in the form :

**"SERVER\user (Long name)"**

Where *SERVER* is the NetBIOS name of the Samba server, *user* is the user name of the UNIX user who owns the file, and *(Long name)* is the discriptive string identifying the user (normally found in the GECOS field of the UNIX password database).

If the parameter *nt acl support* is set to false then the file owner will be shown as the NT user **"Everyone"** and the permissions will be shown as NT "Full Control".

The permissions field is displayed differently for files and directories, so I'll describe the way file permissions are displayed first.

### 8.4.1. File Permissions

The standard UNIX user/group/world triple and the correspinding "read", "write", "execute" permissions triples are mapped by Samba into a three element NT ACL with the 'r', 'w', and 'x' bits mapped into the corresponding NT permissions. The UNIX world permissions are mapped into the global NT group **Everyone**, followed by the list of permissions allowed for UNIX world. The UNIX owner and group permissions are displayed as an NT **user** icon and an NT **local group** icon respectively followed by the list of permissions allowed for the UNIX user and group.

As many UNIX permission sets don't map into common NT names such as **"read"**, **"change"** or **"full control"** then usually the permissions will be prefixed by the words **"Special Access"** in the NT display list.

But what happens if the file has no permissions allowed for a particular UNIX user group or world component ? In order to allow "no permissions" to be seen and modified then Samba overloads the NT **"Take Ownership"** ACL attribute (which has no meaning in UNIX) and reports a component with no permissions as having the NT **"O"** bit set. This was chosen of course to make it look like a zero, meaning zero permissions. More details on the decision behind this will be given below.

### 8.4.2. Directory Permissions

Directories on an NT NTFS file system have two different sets of permissions. The first set of permissions is the ACL set on the directory itself, this is usually displayed in the first set of parentheses in the normal **"RW"** NT style. This first set of permissions is created by Samba in exactly the same way as normal file permissions are, described above, and is displayed in the same way.

The second set of directory permissions has no real meaning in the UNIX permissions world and represents the **"inherited"** permissions that any file created within this directory would inherit.

Samba synthesises these inherited permissions for NT by returning as an NT ACL the UNIX permission mode that a new file created by Samba on this share would receive.

## 8.5. Modifying file or directory permissions

Modifying file and directory permissions is as simple as changing the displayed permissions in the dialog box, and clicking the **OK** button. However, there are limitations that a user needs to be aware of, and also interactions with the standard Samba permission masks and mapping of DOS attributes that need to also be taken into account.

If the parameter *nt acl support* is set to `false` then any attempt to set security permissions will fail with an **"Access Denied"** message.

The first thing to note is that the **"Add"** button will not return a list of users in Samba 2.0.4 (it will give an error message of **"The remote proceedure call failed and did not execute"**). This means that you can only manipulate the current user/group/world permissions listed in the dialog box. This actually works quite well as these are the only permissions that UNIX actually has.

If a permission triple (either user, group, or world) is removed from the list of permissions in the NT dialog box, then when the **"OK"** button is pressed it will be applied as "no permissions" on the UNIX side. If you then view the permissions again the "no permissions" entry will appear as the NT **"O"** flag, as described above. This allows you to add permissions back to a file or directory once you have removed them from a triple component.

As UNIX supports only the "r", "w" and "x" bits of an NT ACL then if other NT security attributes such as "Delete access" are selected then they will be ignored when applied on the Samba server.

When setting permissions on a directory the second set of permissions (in the second set of parentheses) is by default applied to all files within that directory. If this is not what you want you must uncheck the **"Replace permissions on existing files"** checkbox in the NT dialog before clicking **"OK"**.

If you wish to remove all permissions from a user/group/world component then you may either highlight the component and click the **"Remove"** button, or set the component to only have the special **"Take Ownership"** permission (dsplayed as **"O"** ) highlighted.

## 8.6. Interaction with the standard Samba create mask parameters

Note that with Samba 2.0.5 there are four new parameters to control this interaction. These are :

*security mask*

*force security mode*

*directory security mask*

*force directory security mode*

Once a user clicks **"OK"** to apply the permissions Samba maps the given permissions into a user/group/world r/w/x triple set, and then will check the changed permissions for a file against the bits set in the *security mask* parameter. Any bits that were changed that are not set to '1' in this parameter are left alone in the file permissions.

Essentially, zero bits in the *security mask* mask may be treated as a set of bits the user is *not* allowed to change, and one bits are those the user is allowed to change.

If not set explicitly this parameter is set to the same value as the *create mask* parameter to provide compatibility with Samba 2.0.4 where this permission change facility was introduced. To allow a user to modify all the user/group/world permissions on a file, set this parameter to 0777.

Next Samba checks the changed permissions for a file against the bits set in the *force security mode* parameter. Any bits that were changed that correspond to bits set to '1' in this parameter are forced to be set.

Essentially, bits set in the *force security mode* parameter may be treated as a set of bits that, when modifying security on a file, the user has always set to be 'on'.

If not set explicitly this parameter is set to the same value as the *force create mode* parameter to provide compatibility with Samba 2.0.4 where the permission change facility was introduced. To allow a user to modify all the user/group/world permissions on a file, with no restrictions set this parameter to 000.

The *security mask* and *force security mode* parameters are applied to the change request in that order.

For a directory Samba will perform the same operations as described above for a file except using the parameter *directory security mask* instead of *security mask*, and *force directory security mode* parameter instead of *force security mode* .

The *directory security mask* parameter by default is set to the same value as the *directory mask* parameter and the *force directory security mode* parameter by default is set to the same value as the *force directory mode* parameter to provide compatibility with Samba 2.0.4 where the permission change facility was introduced.

In this way Samba enforces the permission restrictions that an administrator can set on a Samba share, whilst still allowing users to modify the permission bits within that restriction.

If you want to set up a share that allows users full control in modifying the permission bits on their files and directories and doesn't force any particular bits to be set 'on', then set the following parameters in the smb.conf(5) file in that share specific section :

*security mask = 0777*

*force security mode = 0*

*directory security mask = 0777*

*force directory security mode = 0*

As described, in Samba 2.0.4 the parameters :

*create mask*

*force create mode*

*directory mask*

*force directory mode*

were used instead of the parameters discussed here.

---

## 8.7. Interaction with the standard Samba file attribute mapping

Samba maps some of the DOS attribute bits (such as "read only") into the UNIX permissions of a file. This means there can be a conflict between the permission bits set via the security dialog and the permission bits set by the file attribute mapping.

One way this can show up is if a file has no UNIX read access for the owner it will show up as "read only" in the standard file attributes tabbed dialog. Unfortunately this dialog is the same one that contains the security info in another tab.

What this can mean is that if the owner changes the permissions to allow themselves read access using the security dialog, clicks **"OK"** to get back to the standard attributes tab dialog, and then clicks **"OK"** on that dialog, then NT will set the file permissions back to read−only (as that is what the attributes still say in the dialog). This means that after setting permissions and clicking **"OK"** to get back to the attributes dialog you should always hit **"Cancel"** rather than **"OK"** to ensure that your changes are not overridden.

---

# Chapter 9. OS2 Client HOWTO

## 9.1. FAQs

### 9.1.1. How can I configure OS/2 Warp Connect or OS/2 Warp 4 as a client for Samba?

A more complete answer to this question can be found on http://carol.wins.uva.nl/~leeuw/samba/warp.html.

Basically, you need three components:

- The File and Print Client ('IBM Peer')
- TCP/IP ('Internet support')
- The "NetBIOS over TCP/IP" driver ('TCPBEUI')

Installing the first two together with the base operating system on a blank system is explained in the Warp manual. If Warp has already been installed, but you now want to install the networking support, use the "Selective Install for Networking" object in the "System Setup" folder.

Adding the "NetBIOS over TCP/IP" driver is not described in the manual and just barely in the online documentation. Start MPTS.EXE, click on OK, click on "Configure LAPS" and click on "IBM OS/2 NETBIOS OVER TCP/IP" in 'Protocols'. This line is then moved to 'Current Configuration'. Select that line, click on "Change number" and increase it from 0 to 1. Save this configuration.

If the Samba server(s) is not on your local subnet, you can optionally add IP names and addresses of these servers to the "Names List", or specify a WINS server ('NetBIOS Nameserver' in IBM and RFC terminology). For Warp Connect you may need to download an update for 'IBM Peer' to bring it on the same level as Warp 4. See the webpage mentioned above.

---

### 9.1.2. How can I configure OS/2 Warp 3 (not Connect), OS/2 1.2, 1.3 or 2.x for Samba?

You can use the free Microsoft LAN Manager 2.2c Client for OS/2 from ftp://ftp.microsoft.com/BusSys/Clients/LANMAN.OS2/. See http://carol.wins.uva.nl/~leeuw/lanman.html for more information on how to install and use this client. In a nutshell, edit the file \OS2VER in the root directory of the OS/2 boot partition and add the lines:

```
20=setup.exe
20=netwksta.sys
20=netvdd.sys
```

before you install the client. Also, don't use the included NE2000 driver because it is buggy. Try the NE2000 or NS2000 driver from ftp://ftp.cdrom.com/pub/os2/network/ndis/ instead.

---

## 9.1.3. Are there any other issues when OS/2 (any version) is used as a client?

When you do a NET VIEW or use the "File and Print Client Resource Browser", no Samba servers show up. This can be fixed by a patch from http://carol.wins.uva.nl/~leeuw/samba/fix.html. The patch will be included in a later version of Samba. It also fixes a couple of other problems, such as preserving long filenames when objects are dragged from the Workplace Shell to the Samba server.

---

## 9.1.4. How do I get printer driver download working for OS/2 clients?

First, create a share called [PRINTDRV] that is world−readable. Copy your OS/2 driver files there. Note that the .EA_ files must still be separate, so you will need to use the original install files, and not copy an installed driver from an OS/2 system.

Install the NT driver first for that printer. Then, add to your smb.conf a paramater, "os2 driver map = *filename*". Then, in the file specified by *filename*, map the name of the NT driver name to the OS/2 driver name as follows:

<nt driver name> = <os2 driver name>.<device name>, e.g.: HP LaserJet 5L = LASERJET.HP LaserJet 5L

You can have multiple drivers mapped in this file.

If you only specify the OS/2 driver name, and not the device name, the first attempt to download the driver will actually download the files, but the OS/2 client will tell you the driver is not available. On the second attempt, it will work. This is fixed simply by adding the device name to the mapping, after which it will work on the first attempt.