8 Additional Samba Information

This chapter wraps up our coverage of the *smb.conf* configuration file with some miscellaneous options that can perform a variety of tasks. We will talk briefly about options for supporting programmers, internationalization, messages, and common Windows bugs. For the most part, you will use these options only in isolated circumstances. We also cover performing automated backups with the *smbtar* command at the end of this chapter. So without further ado, let's jump into our first subject: options to help programmers.

Supporting Programmers

If you have programmers accessing your Samba server, you'll want to be aware of the special options listed in Table 8-1.

Table 8-1. Programming Configuration Options

Option	Parameters	Function	Default	Scope
`time server`	boolean	If `yes`, *nmbd* announces itself as a SMB time service to Windows clients.	`no`	Global
`time offset`	numerical (number of minutes)	Adds a specified number of minutes to the reported time.	`0`	Global
`dos filetimes`	boolean	Allows non-owners of a file to change its time if they can write to it.	`no`	Share
`dos filetime resolution`	boolean	Causes file times to be rounded to the next even second.	`no`	Share
`fake directory create times`	boolean	Sets directory times to avoid a MS *nmake* bug.	`no`	Share

Time Synchronization

Time synchronization can be very important to programmers. Consider the following options:

```
time service = yes
dos filetimes = yes
fake directory create times = yes
dos filetime resolution = yes
delete readonly = yes
```

If you set these options, Samba shares will provide the kind of compatible file times that Visual C++, *nmake*, and other Microsoft programming tools require. Otherwise, PC *make* programs will tend to think that all the files in a directory need to be recompiled every time. Obviously, this is not the behavior you want.

time server

If your Samba server has an accurate clock, or if it's a client of one of the Unix network time servers, you can instruct it to advertise itself as an SMB time server by setting the `time server` option as follows:

```
[global]
    time service = yes
```

The client will still have to request the correct time with the following DOS command, substituting the Samba server name in at the appropriate point:

```
C:\NET TIME \\server /YES /SET
```

This command can be placed in a Windows logon script (see Chapter 6, *Users, Security, and Domains*).

By default, the `time server` option is normally set to `no`. If you turn this service on, you can use the command above to keep the client clocks from drifting. Time synchronization is important to clients using programs such as *make*, which compile based on the last time the file was changed. Incorrectly synchronized times can cause such programs to either remake all files in a directory, which wastes time, or not recompile a source file that was just modified because of a slight clock drift.

time offset

To deal with clients that don't process daylight savings time properly, Samba provides the `time offset` option. If set, it adds the specified number of minutes to the current time. This is handy if you're in Newfoundland and Windows doesn't know about the 30-minute time difference there:

```
[global]
    time offset = 30
```

dos filetimes

Traditionally, only the root user and the owner of a file can change its last-modified date on a Unix system. The share-level `dos filetimes` option allows the Samba server to mimic the characteristics of a DOS/Windows machine: any user can change the last modified date on a file in that share if he or she has write permission to it. In order to do this, Samba uses its root privileges to modify the timestamp on the file.

By default, this option is disabled. Setting this option to `yes` is often necessary to allow PC *make* programs to work properly. Without it, they cannot change the last-modified date themselves. This often results in the program thinking *all* files need recompiling when they really don't.

dos filetime resolution

`dos filetime resolution` is share-level option. If set to `yes`, Samba will arrange to have the file times rounded to the closest two-second boundary. This option exists primarily to satisfy a quirk in Windows that prevents Visual C++ from correctly recognizing that a file has not changed. You can enable it as follows:

```
[data]
    dos filetime resolution = yes
```

We recommend using this option only if you are using Microsoft Visual C++ on a Samba share that supports opportunistic locking.

fake directory create times

The `fake directory create times` option exists to keep PC *make* programs sane. VFAT and NTFS filesystems record the creation date of a specific directory while Unix does not. Without this option, Samba takes the earliest recorded date it has for the directory (often the last-modified date of a file) and returns it to the client. If this is not sufficient, set the following option under a share definition:

```
[data]
    fake directory create times = yes
```

If set, Samba will adjust the directory create time it reports to the hardcoded value January 1st, 1980. This is primarily used to convince the Visual C++ *nmake* program that any object files in its build directories are indeed younger than the creation date of the directory itself and need to be recompiled.

Magic Scripts

The following options deal with *magic scripts* on the Samba server. Magic scripts are a method of running programs on Unix and redirecting the output back to the SMB client. These are essentially an experimental hack. However, some users and their programs still rely on these two options for their programs to function correctly. Magic scripts are not widely trusted and their use is highly discouraged by the Samba team. See Table 8-2 for more information.

Table 8-2. Networking Configuration Options

Option	Parameters	Function	Default	Scope
`magic script`	string (fully-qualified filename)	Sets the name of a file to be executed by Samba, as the logged-on user, when closed.	None	Share
`magic output`	string (fully-qualified filename)	Sets a file to log output from the magic file.	*scriptname.out*	Share

magic script

If the `magic script` option is set to a filename and the client creates a file by that name in that share, Samba will run the file as soon as the user has opened and closed it. For example, let's assume that the following option was created in the share `[accounting]`:

```
[accounting]
    magic script = tally.sh
```

Samba continually monitors the files in that share. If one by the name of *tally.sh* is closed (after being opened) by a user, Samba will execute the contents of that file locally. The file will be passed to the shell to execute; it must therefore be a legal Unix shell script. This means that it must have newline characters as line endings instead of Windows CR/LFs. In addition, it helps if you use the `#!` directive at the beginning of the file to indicate under which shell the script should run.

magic output

This option specifies an output file that the script specified by the `magic script` option will send output to. You must specify a filename in a writable directory:

```
[accounting]
    magic script = tally.sh
    magic output = /var/log/magicoutput
```

If this option is omitted, the default output file is the name of the script (as stated in the `magic script` option) with the extension *.out* appended onto it.

Internationalization

Samba has a limited ability to speak foreign tongues: if you need to deal with characters that aren't in standard ASCII, some options that can help you are shown in Table 8-3. Otherwise, you can skip over this section.

Table 8-3. Networking Configuration Options

Option	Parameters	Function	Default	Scope
`client code page`	Described in this section	Sets a code page to expect from clients	850	Global
`character set`	Described in this section	Translates code pages into alternate UNIX character sets	None	Global
`coding system`	Described in this section	Translates code page 932 into an Asian character set	None	Global
`valid chars`	string (set of characters)	Obsolete: formerly added individual characters to a code page, and had to be used after setting client code page	None	Global

client code page

The character sets on Windows platforms hark back to the original concept of a *code page*. These code pages are used by DOS and Windows clients to determine rules for mapping lowercase letters to uppercase letters. Samba can be instructed to use a variety of code pages through the use of the global `client code page` option in order to match the corresponding code page in use on the client. This option loads a code-page definition file, and can take the values specified in Table 8-4.

Table 8-4. Valid Code Pages with Samba 2.0

Code Page	Definition
437	MS-DOS Latin (United States)
737	Windows 95 Greek
850	MS-DOS Latin 1 (Western European)
852	MS-DOS Latin 2 (Eastern European)
861	MS-DOS Icelandic
866	MS-DOS Cyrillic (Russian)
932	MS-DOS Japanese Shift-JIS
936	MS-DOS Simplified Chinese
949	MS-DOS Korean Hangul
950	MS-DOS Traditional Chinese

You can set the client code page as follows:

```
[global]
    client code page = 852
```

The default value of this option is 850. You can use the *make_smbcodepage* tool that comes with Samba (by default in */usr/local/samba/bin*) to create your own SMB code pages, in the event that those listed earlier are not sufficient.

character set

The global `character set` option can be used to convert filenames offered through a DOS code page (see the previous section, "client code page") to equivalents that can be represented by Unix character sets other than those in the United States. For example, if you want to convert the Western European MS-DOS character set on the client to a Western European Unix character set on the server, you can use the following in your configuration file:

```
[global]
    client code page = 850
    character set = ISO8859-1
```

Note that you must include a `client code page` option to specify the character set from which you are converting. The valid character sets (and their matching code pages) that Samba 2.0 accepts are listed in Table 8-5:

Table 8-5. Valid Character Sets with Samba 2.0

Character Set	Matching Code Page	Definition
`ISO8859-1`	850	Western European Unix
`ISO8859-2`	852	Eastern European Unix
`ISO8859-5`	866	Russian Cyrillic Unix
`KOI8-R`	866	Alternate Russian Cyrillic Unix

Normally, the `character set` option is disabled completely.

coding system

The `coding system` option is similar to the `character set` option. However, its purpose is to determine how to convert a Japanese Shift JIS code page into an appropriate Unix character set. In order to use this option, the `client code page` option described previously must be set to page 932. The valid coding systems that Samba 2.0 accepts are listed in Table 8-6.

Table 8-6. Valid Coding System Parameters with Samba 2.0

Character Set	Definition
`SJIS`	Standard Shift JIS
`JIS8`	Eight-bit JIS codes

Table 8-6. Valid Coding System Parameters with Samba 2.0 (continued)

Character Set	Definition
`J8BB`	Eight-bit JIS codes
`J8BH`	Eight-bit JIS codes
`J8@B`	Eight-bit JIS codes
`J8@J`	Eight-bit JIS codes
`J8@H`	Eight-bit JIS codes
`JIS7`	Seven-bit JIS codes
`J7BB`	Seven-bit JIS codes
`J7BH`	Seven-bit JIS codes
`J7@B`	Seven-bit JIS codes
`J7@J`	Seven-bit JIS codes
`J7@H`	Seven-bit JIS codes
`JUNET`	JUNET codes
`JUBB`	JUNET codes
`JUBH`	JUNET codes
`JU@B`	JUNET codes
`JU@J`	JUNET codes
`JU@H`	JUNET codes
`EUC`	EUC codes
`HEX`	Three-byte hexidecimal code
`CAP`	Three-byte hexidecimal code (Columbia Appletalk Program)

valid chars

The `valid chars` option is an older Samba feature that will add individual characters to a code page. However, this option is being phased out in favor of more modern coding systems. You can use this option as follows:

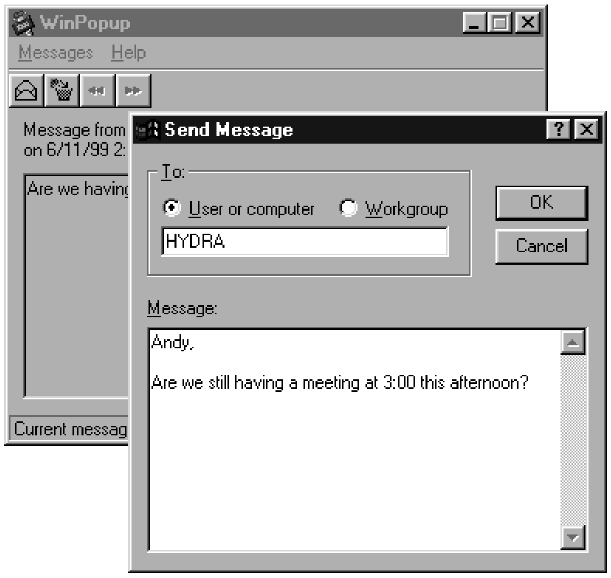
```
valid chars = Î
valid chars = 0450:0420 0x0A20:0x0A00
valid chars = A:a
```

Each of the characters in the list specified should be separated by spaces. If there is a colon between two characters or their numerical equivalents, the data to the left of the colon is considered an uppercase character, while the data to the right is considered the lowercase character. You can represent characters both by literals (if you can type them) and by octal, hexidecimal, or decimal Unicode equivalents.

We recommend against using this option. Instead, go with one of the standard code pages listed earlier in this section. If you do use this option, however, it must be listed after the `client code page` to which you wish to add the character. Otherwise, the characters will not be added.

WinPopup Messages

You can use the WinPopup tool (*WINPOPUP.EXE*) in Windows to send messages to users, machines, or entire workgroups on the network. This tool is provided with Windows 95 OSR2 and comes standard with Windows 98. With either Windows 95 or 98, however, you need to be running WinPopup to receive and send WinPopup messages. With Windows NT, you can still receive messages without starting such a tool; they will automatically appear in a small dialog box on the screen when received. The WinPopup application is shown in Figure 8-1.



HYDRA

Andy,

Are we still having a meeting at 3:00 this afternoon?

Figure 8-1. The WinPopup application

Samba has a single WinPopup messaging option, `message command`, as shown in Table 8-7.

Table 8-7. WinPopup Configuration Option

Option	Parameter	Function	Default	Scope
`message command`	string (fully-qualified pathname)	Sets a command to run on Unix when a WinPopup message is received.	None	Global

message command

Samba's `message command` option sets the path to a program that will run on the server when a Windows popup message arrives at the server. The command will be executed using the `guest account` user. What to do with one of these is questionable since it's probably for the Samba administrator, and Samba doesn't know his or her name. If you know there's a human using the console, the Samba team once suggested the following:

```
[global]
    message command = /bin/csh -c 'xedit %s; rm %s' &
```

Note the use of variables here. The `%s` variable will become the file that the message is in. This file should be deleted when the command is finished with it; otherwise, there will be a buildup of pop-up files collecting on the Samba server. In addition, the command must fork its own process (note the & after the command); otherwise the client may suspend and wait for notification that the command was sent successfully before continuing.

In addition to the standard variables, Table 8-8 shows the three unique variables that you can use in a `message command`.

Table 8-8. Message Command Variables

Variable	Definition
`%s`	The name of the file in which the message resides
`%f`	The name of the client that sent the message
`%t`	The name of the machine that is the destination of the message

Recently Added Options

Samba has several options that appeared around the time of Samba 2.0, but are not entirely supported. However, we will give you a brief overview of their workings in this section. These options are shown in Table 8-9.

Table 8-9. Recently Added Options

Option	Parameters	Function	Default	Scope
`change notify timeout`	numerical (number of seconds)	Sets the interval between checks when a client asks to wait for a change in a specified directory.	`60`	Global
`machine password timeout`	numerical (number of seconds)	Sets the renewal interval for NT domain machine passwords.	`604,800` (1 week)	Global

Table 8-9. Recently Added Options (continued)

Option	Parameters	Function	Default	Scope
`stat cache`	boolean	If `yes`, Samba will cache recent name mappings.	`yes`	Global
`stat cache size`	numerical	Sets the size of the stat cache.	`50`	Global

change notify timeout

The `change notify timeout` global option emulates a Windows NT SMB feature called *change notification*. This allows a client to request that a Windows NT server periodically monitor a specific directory on a share for any changes. If any changes occur, the server will notify the client.

As of version 2.0, Samba will perform this function for its clients. However, performing these checks too often can slow the server down considerably. This option sets the time period that Samba should wait between such checks. The default is one minute (60 seconds); however, you can use this option to specify an alternate time that Samba should wait between performing checks:

```
[global]
    change notify timeout = 30
```

machine password timeout

The `machine password timeout` global option sets a retention period for NT domain machine passwords. The default is currently set to the same time period that Windows NT 4.0 uses: 604,800 seconds (one week). Samba will periodically attempt to change the *machine account password*, which is a password used specifically by another server to report changes to it. This option specifies the number of seconds that Samba should wait before attempting to change that password. The following example changes it to a single day, by specifying the following:

```
[global]
    machine password timeout = 86400
```

stat cache

The `stat cache` global option turns on caching of recent case-insensitive name mappings. The default is `yes`. The Samba team recommends that you never change this parameter.

stat cache size

The `stat cache size` global option sets the size of the cache entries to be used for the `stat cache` option. The default here is 50. Again, the Samba team recommends that you never change this parameter.

Miscellaneous Options

Many Samba options are present to deal with operating system issues on either Unix or Windows. The options shown in Table 8-10 deal specifically with some of these known problems. We usually don't change these and we recommend the same to you.

Table 8-10. Miscellaneous Options

Option	Parameters	Function	Default	Scope
`deadtime`	numerical (number of minutes)	Specifies the number of minutes of inactivity before a connection should be terminated.	`0`	Global
`dfree command`	string (command)	Used to provide a command that returns disk free space in a format recognized by Samba.	None	Global
`fstype`	`NTFS`, `FAT`, or `Samba`	Sets the filesystem type reported by the server to the client.	`NTFS`	Global
`keep alive`	seconds	Sets the number of seconds between checks for an inoperative client.	0 (none)	Global
`max disk size`	numerical (size in MB)	Sets the largest disk size to return to a client, some of which have limits. Does not affect actual operations on the disk.	0 (infinity)	Global
`max mux`	numerical	Sets the maximum number of simultaneous SMB operations that clients may make.	`50`	Global
`max open files`	numerical	Limits number of open files to be below Unix limits.	`10,000`	Global
`max xmit`	numerical	Specifies the maximum packet size that Samba will send.	`65,535`	Global
`nt pipe support`	boolean	Turns off an experimental NT feature, for benchmarking or in case of an error.	`yes`	Global
`nt smb support`	boolean	Turns off an experimental NT feature, for benchmarking or in case of an error.	`yes`	Global
`ole locking compatibility`	boolean	Remaps out-of-range lock requests used on Windows to fit in allowable range on Unix. Turning it off causes Unix lock errors.	`yes`	Global
`panic action`	command	Program to run if Samba server fails; for debugging.	None	Global
`set directory`	boolean	If `yes`, allows VMS clients to issue `set dir` commands.	`no`	Global

Table 8-10. Miscellaneous Options (continued)

Option	Parameters	Function	Default	Scope
`smbrun`	string (fully-qualified command)	Sets the command Samba uses as a wrapper for shell commands.	None	Global
`status`	boolean	If `yes`, allows Samba to monitor status for `smbstatus` command.	`yes`	Global
`strict sync`	boolean	If `no`, ignores Windows applications requests to perform a sync-to-disk.	`no`	Global
`sync always`	boolean	If `yes`, forces all client writes to be committed to disk before returning from the call.	`no`	Global
`strip dot`	boolean	If `yes`, strips trailing dots from Unix filenames.	`no`	Global

deadtime

This global option sets the number of minutes that Samba will wait for an inactive client before closing its session with the Samba server. A client is considered inactive when it has no open files and there is no data being sent from it. The default value for this option is 0, which means that Samba never closes any connections no matter how long they have been inactive. You can override it as follows:

```
[global]
    deadtime = 10
```

This tells Samba to terminate any inactive client sessions after 10 minutes. For most networks, setting this option as such will work because reconnections from the client are generally performed transparently to the user.

dfree command

This global option is used on systems that incorrectly determine the free space left on the disk. So far, the only confirmed system that needs this option set is Ultrix. There is no default value for this option, which means that Samba already knows how to compute the free disk space on its own and the results are considered reliable. You can override it as follows:

```
[global]
    dfree command = /usr/local/bin/dfree
```

This option should point to a script that should return the total disk space in a block, and the number of available blocks. The Samba documentation recommends the following as a usable script:

```
#!/bin/sh
df $1 | tail -1 | awk '{print $2" "$4}'
```

On System V machines, the following will work:

```
#!/bin/sh
/usr/bin/df $1 | tail -1 | awk '{print $3" "$5}'
```

fstype

This share-level option sets the type of filesystem that Samba reports when queried by the client. There are three strings that can be used as a value to this configuration option, as listed in Table 8-11.

Table 8-11. Filesystem Types

Variable	Definition
NTFS	Microsoft Windows NT filesystem
FAT	DOS FAT filesystem
Samba	Samba filesystem

The default value for this option is `NTFS`, which represents a Windows NT filesystem. There probably isn't a need to specify any other type of filesystem. However, if you need to, you can override it per share as follows:

```
[data]
    fstype = FAT
```

keep alive

This global option specifies the number of seconds that Samba waits between sending NetBIOS *keep-alive packets*. These packets are used to ping a client to detect whether it is still alive and on the network. The default value for this option is `0`, which means that Samba will not send any such packets at all. You can override it as follows:

```
[global]
    keep alive = 10
```

max disk size

This global option specifies an illusory limit, in megabytes, for each of the shares that Samba is using. You would typically set this option to prevent clients with older operating systems from incorrectly processing large disk spaces, such as those over one gigabyte.

The default value for this option is `0`, which means there is no upper limit at all. You can override it as follows:

```
[global]
    max disk size = 1000
```

max mux

This global option specifies the maximum number of concurrent SMB operations that Samba allows. The default value for this option is `50`. You can override it as follows:

```
[global]
    max mux = 100
```

max open files

This global option specifies the maximum number of open files that Samba should allow at any given time for all processes. This value must be equal to or less than the amount allowed by the operating system, which varies from system to system. The default value for this option is `10,000`. You can override it as follows:

```
[global]
    max open files = 8000
```

max xmit

This global option sets the maximum size of packets that Samba exchanges with a client. In some cases, setting a smaller maximum packet size can increase performance, especially with Windows for Workgroups. The default value for this option is `65535`. You can override it as follows:

```
[global]
    max xmit = 4096
```

The section "The TCP receive window," in Appendix B, *Samba Performance Tuning*," shows some uses for this option.

nt pipe support

This global option is used by developers to allow or disallow Windows NT clients the ability to make connections to the NT SMB-specific IPC$ pipes. As a user, you should never need to override the default:

```
[global]
    nt pipe support = yes
```

nt smb support

This global option is used by developers to negotiate NT-specific SMB options with Windows NT clients. The Samba team has discovered that slightly better performance comes from setting this value to `no`. However, as a user, you should probably not override the default:

```
[global]
    nt smb support = yes
```

ole locking compatibility

This global option turns off Samba's internal byte-range locking manipulation in files, which gives compatibility with Object Linking and Embedding (OLE) applications that use high byte-range locks as a method of interprocess communication. The default value for this option is `yes`. If you trust your Unix locking mechanisms, you can override it as follows:

```
[global]
    ole locking compatibility = no
```

panic action

This global option specifies a command to execute in the event that Samba itself encounters a fatal error when loading or running. There is no default value for this option. You can specify an action as follows:

```
[global]
    panic action = /bin/csh -c
          'xedit < "Samba has shutdown unexpectedly!'
```

set directory

This boolean share-level option allows Digital Pathworks clients to use the `setdir` command to change directories on the server. If you are not using the Digital Pathworks client, you should not need to alter this option. The default value for this option is `no`. You can override it per share as follows:

```
[data]
    set directory = yes
```

smbrun

This option sets the location of the *smbrun* executable, which Samba uses as a wrapper to run shell commands. The default value for this option is automatically configured by Samba when it is compiled. If you did not install Samba to the standard directory, you can specify where the binary is as follows:

```
[global]
    smbrun = /usr/local/bin/smbrun
```

status

This global option indicates whether Samba should log all active connections to a status file. This file is used only by the *smbstatus* command. If you have no intentions of using this command, you can set this option to `no`, which can result in a small increase of speed on the server. The default value for this option is `yes`. You can override it as follows:

```
[global]
    status = no
```

strict sync

This share-level option determines whether Samba honors all requests to perform a disk sync when requested to do so by a client. Many clients request a disk sync when they are really just trying to flush data to their own open files. As a result, this can substantially slow a Samba server down. The default value for this option is `no`. You can override it as follows:

```
[data]
    strict sync = yes
```

sync always

This share-level option decides whether every write to disk should be followed by a disk synchronization before the write call returns control to the client. Even if the value of this option is `no`, clients can request a disk synchronization; see the `strict sync` option above. The default value for this option is `no`. You can override it per share as follows:

```
[data]
    sync always = yes
```

strip dot

This global option determines whether to remove the trailing dot from Unix filenames that are formatted with a dot at the end. The default value for this option is `no`. You can override it per share as follows:

```
[global]
    strip dot = yes
```

This option is now considered obsolete; the user should use the `mangled map` option insead.

Backups with smbtar

Our final topic in this chapter is the *smbtar* tool. One common problem with modem PCs is that floppies and even CD-ROMs are often too small to use for backups. However, buying one tape drive per machine would also be silly.

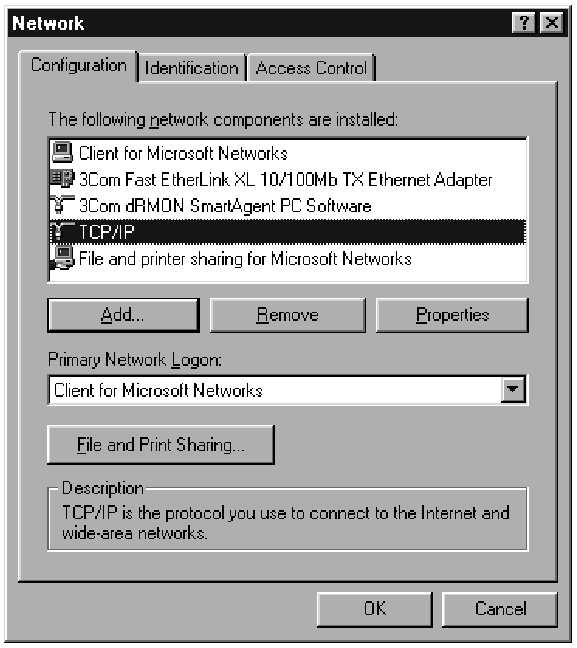
Consequently, many sites don't back up their PCs at all. Instead, they reinstall them using floppy disks and CD-ROMs when they fail.

Thankfully, Samba provides us with another option: you can back up PCs' data using the *smbtar* tool. This can be done on a regular basis if you keep user data on your Samba system, or only occasionally, to save the local applications and configuration files and thus make repairs and reinstallations quicker.

To back up PCs from a Unix server, you need to do three things:

1. Ensure that File and Printer Sharing is installed on the PC and is bound to the TCP/IP protocol.
2. Explicitly share a disk on the PC so it can be read from the server.
3. Set up the backup scripts on the server.

We'll use Windows 95/98 to illustrate the first two steps. Go to the Networking icon in the Control Panel window, and check that File and Printer Sharing for Microsoft Networks is currently listed in the top window, as shown in Figure 8-2.



The following network components are installed:
Client for Microsoft Networks
3Com Fast EtherLink XL 10/100Mb TX Ethernet Adapter
3Com dRMON SmartAgent PC Software
TCP/IP
File and printer sharing for Microsoft Networks

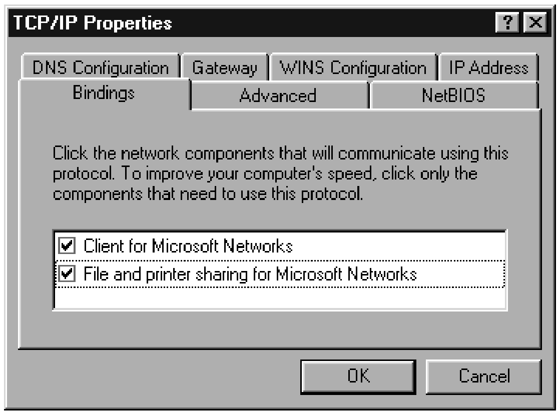
Primary Network Logon:
Client for Microsoft Networks

Description
TCP/IP is the protocol you use to connect to the Internet and wide-area networks.

Figure 8-2. The Networking window

If “File and printer sharing for Microsoft Networks” isn’t installed, you can install it by clicking on the Add button on the Network panel. After pressing it, you will be asked what service to add. Select Service and move forward, and you will be asked for a vendor and a service to install. Finally, select “File and printer sharing for Microsoft Networks,” and click on Done to install the service.

Once you’ve installed “File and printer sharing for Microsoft Networks,” return to the Network panel and select the TCP/IP protocol that is tied to your Samba network adapter. Then, click on the Properties button and choose the Bindings tab at the top. You should see a dialog box similar to Figure 8-3. Here, you’ll need to verify that the “File and Printer Sharing” checkbox is checked, giving it access to TCP/IP. At this point you can share disks with other machines on the net.



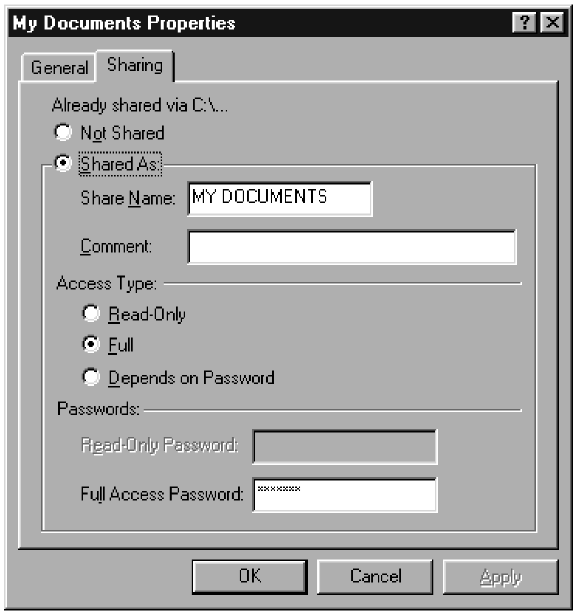
Click the network components that will communicate using this protocol. To improve your computer's speed, click only the components that need to use this protocol.

- ✓ Client for Microsoft Networks
- ✓ File and printer sharing for Microsoft Networks

Figure 8-3. TCP/IP Bindings

The next step is to share the disk you want to back up with the tape server. Go to My Computer and select, for example, the My Documents directory. Then right-click on the icon and select its Properties. This should yield the dialog box in Figure 8-4.

Select the Sharing tab and turn file sharing on. You now have the choice to share the disk as read-only, read-write (Full), or either, each with separate password. This is the Windows 95/98 version, so it provides only share-level security. In this example, we made it read/write and set a password, as shown in Figure 8-5. When you enter the password and click on OK, you’ll be prompted to re-enter it. After that, you have finished the second step.



Already shared via C:\...

Not Shared [✓ Shared As:]

Share Name: MY DOCUMENTS

Comment:

Access Type: Read-Only [✓ Full] Depends on Password

Passwords:

Read-Only Password:

Full Access Password: *******

Figure 8-4. My Documents Properties

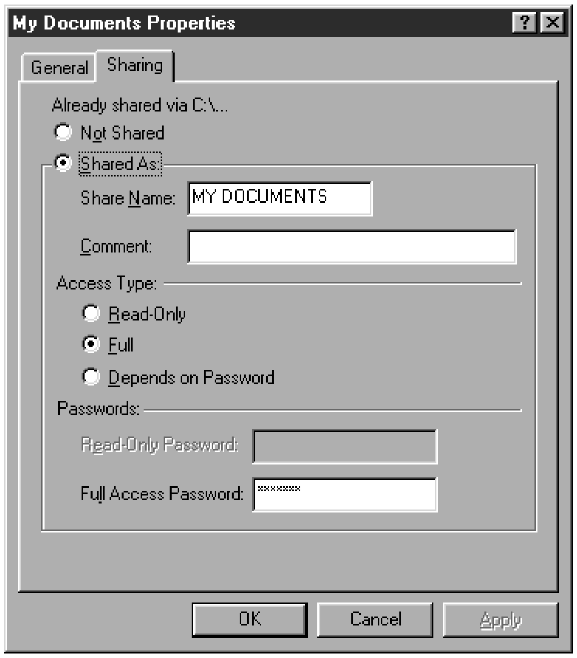
Finally, the last step is to set up a backup script on the tape server, using the *smbtar* program. The simplest script might contain only a single line and would be something like the following:

```
smbtar -s client -t /dev/rst0 -x "My Documents" -p password
```

This unconditionally backs up the *//client/My Documents* share to the device */dev/rst0*. Of course, this is excessively simple and quite insecure. What you will want to do will depend on your existing backup scheme.

However, to whet your appetite, here are some possibilities of what *smbtar* can do:

- Back up files incrementally using the DOS archive bit (the `-i` option). This requires the client share to be accessed read-write so the bit can be cleared by *smbtar*
- Back up only files that have changed since a specified date (using the `-N` ***`filename`*** option)
- Back up entire PC drives, by sharing all of C: or D:, for example, and backing that up



My Documents Properties

General | Sharing

Already shared via C:\...

Not Shared [✓ Shared As:]

Share Name: MY DOCUMENTS

Comment:

Access Type:

Read-Only [✓ Full] Depends on Password

Passwords:

Read-Only Password:

Full Access Password: ********

Figure 8-5. MyFiles Properties as shared

Except for the first example, each of these can be done with the PC sharing set to read-only, reducing the security risk of having passwords in scripts and passing them on the command line.