

**NAME**

smbclient - ftp-like client to access SMB/CIFS resources on servers

**SYNOPSIS**

**smbclient** [-b <buffer size>] [-d debuglevel] [-L <netbios name>] [-U username] [-I destinationIP] [-M <netbios name>] [-m maxprotocol] [-A authfile] [-N] [-i scope] [-O <socket options>] [-p port] [-R <name resolve order>] [-s <smb config file>] [-k] [-P] [-c <command>]

**smbclient** {servicename} [password] [-b <buffer size>] [-d debuglevel] [-D Directory] [-U username] [-W workgroup] [-M <netbios name>] [-m maxprotocol] [-A authfile] [-N] [-l logdir] [-I destinationIP] [-E] [-c <command string>] [-i scope] [-O <socket options>] [-p port] [-R <name resolve order>] [-s <smb config file>] [-T<c|x>IXFqgbNan] [-k]

**DESCRIPTION**

This tool is part of the **samba(7)** suite.

**smbclient** is a client that can 'talk' to an SMB/CIFS server. It offers an interface similar to that of the ftp program (see **ftp(1)**). Operations include things like getting files from the server to the local machine, putting files from the local machine to the server, retrieving directory information from the server and so on.

**OPTIONS**

## servicename

servicename is the name of the service you want to use on the server. A service name takes the form *//server/service* where *server* is the NetBIOS name of the SMB/CIFS server offering the desired service and *service* is the name of the service offered. Thus to connect to the service "printer" on the SMB/CIFS server "smbserver", you would use the servicename *//smbserver/printer*

Note that the server name required is NOT necessarily the IP (DNS) host name of the server ! The name required is a NetBIOS server name, which may or may not be the same as the IP hostname of the machine running the server.

The server name is looked up according to either the *-R* parameter to **smbclient** or using the name resolve order parameter in the **smb.conf(5)** file, allowing an administrator to change the order and methods by which server names are looked up.

## password

The password required to access the specified service on the specified server. If this parameter is supplied, the *-N* option (suppress password prompt) is assumed.

There is no default password. If no password is supplied on the command line (either by using this parameter or adding a password to the *-U* option (see below)) and the *-N* option is not specified, the client will prompt for a password, even if the desired service does not require one. (If no password is required, simply press ENTER to provide a null password.)

Note: Some servers (including OS/2 and Windows for Workgroups) insist on an uppercase password. Lowercase or mixed case passwords may be rejected by these servers.

Be cautious about including passwords in scripts.

*-R* <name resolve order>

This option is used by the programs in the Samba suite to determine what naming services and in what order to resolve host names to IP addresses. The option takes a space-separated string of different name resolution options.

The options are : "lmhosts", "host", "wins" and "bcast". They cause names to be resolved as follows:

&#8226;

**lmhosts:** Lookup an IP address in the Samba `lmhosts` file. If the line in `lmhosts` has no name type attached to the NetBIOS name (see the **lmhosts(5)** for details) then any name type matches for lookup.

&#8226;

**host:** Do a standard host name to IP address resolution, using the system `/etc/hosts`, NIS, or DNS lookups. This method of name resolution is operating system dependent, for instance on IRIX or Solaris this may be controlled by the `/etc/nsswitch.conf` file). Note that this method is only used if the NetBIOS name type being queried is the `0x20` (server) name type, otherwise it is ignored.

&#8226;

**wins:** Query a name with the IP address listed in the `wins server` parameter. If no WINS server has been specified this method will be ignored.

&#8226;

**bcast:** Do a broadcast on each of the known local interfaces listed in the `interfaces` parameter. This is the least reliable of the name resolution methods as it depends on the target host being on a locally connected subnet.

If this parameter is not set then the name resolve order defined in the **smb.conf(5)** file parameter (name resolve order) will be used.

The default order is `lmhosts`, `host`, `wins`, `bcast` and without this parameter or any entry in the `name resolve order` parameter of the **smb.conf(5)** file the name resolution methods will be attempted in this order.

#### -M NetBIOS name

This options allows you to send messages, using the "WinPopup" protocol, to another computer. Once a connection is established you then type your message, pressing `^D` (control-D) to end.

If the receiving computer is running WinPopup the user will receive the message and probably a beep. If they are not running WinPopup the message will be lost, and no error message will occur.

The message is also automatically truncated if the message is over 1600 bytes, as this is the limit of the protocol.

One useful trick is to cat the message through **smbclient**. For example:

```
cat mymessage.txt | smbclient -M FRED
```

will send the message in the file `mymessage.txt` to the machine FRED.

You may also find the `-U` and `-I` options useful, as they allow you to control the FROM and TO parts of the message.

See the `message command` parameter in the **smb.conf(5)** for a description of how to handle incoming WinPopup messages in Samba.

**Note:** Copy WinPopup into the startup group on your WfWg PCs if you want them to always be able to receive messages.

#### -p port

This number is the TCP port number that will be used when making connections to the server. The standard (well-known) TCP port number for an SMB/CIFS server is 139, which is the default.

-P Make queries to the external server using the machine account of the local server.

-h|--help

Print a summary of command line options.

-I IP-address

*IP address* is the address of the server to connect to. It should be specified in standard "a.b.c.d" notation.

Normally the client would attempt to locate a named SMB/CIFS server by looking it up via the NetBIOS name resolution mechanism described above in the *name resolve order* parameter above. Using this parameter will force the client to assume that the server is on the machine with the specified IP address and the NetBIOS name component of the resource being connected to will be ignored.

There is no default for this parameter. If not supplied, it will be determined automatically by the client as described above.

-E This parameter causes the client to write messages to the standard error stream (stderr) rather than to the standard output stream.

By default, the client writes messages to standard output - typically the user's tty.

-L This option allows you to look at what services are available on a server. You use it as **smbclient -L host** and a list should appear. The *-I* option may be useful if your NetBIOS names don't match your TCP/IP DNS host names or if you are trying to reach a host on another network.

-t terminal code

This option tells **smbclient** how to interpret filenames coming from the remote server. Usually Asian language multibyte UNIX implementations use different character sets than SMB/CIFS servers (**EUC** instead of

**SJIS** for example). Setting this parameter will let **smbclient** convert between the UNIX filenames and the SMB filenames correctly. This option has not been seriously tested and may have some problems.

The terminal codes include CWsjis, CWeuc, CWjis7, CWjis8, CWjunet, CWhex, CWcap. This is not a complete list, check the Samba source code for the complete list.

-b buffersize

This option changes the transmit/send buffer size when getting or putting a file from/to the server. The default is 65520 bytes. Setting this value smaller (to 1200 bytes) has been observed to speed up file transfers to and from a Win9x server.

-V

Prints the program version number.

-s <configuration file>

The file specified contains the configuration details required by the server. The information in this file includes server-specific information such as what printcap file to use, as well as descriptions of all the services that the server is to provide. See *smb.conf* for more information. The default configuration file name is determined at compile time.

-d|--debuglevel=level

*level* is an integer from 0 to 10. The default value if this parameter is not specified is zero.

The higher this value, the more detail will be logged to the log files about the activities of the server. At level 0, only critical errors and serious warnings will be logged. Level 1 is a reasonable level for day-to-day running - it generates a small amount of information about operations carried out.

Levels above 1 will generate considerable amounts of log data, and should only be used when investigating a problem. Levels above 3 are designed for use only by developers and generate HUGE amounts of log data, most of which is extremely cryptic.

Note that specifying this parameter here will override the

parameter in the *smb.conf* file.

**-l|--logfile=logdirectory**

Base directory name for log/debug files. The extension "**.progname**" will be appended (e.g. log.smbclient, log.smbd, etc...). The log file is never removed by the client.

**-N**

If specified, this parameter suppresses the normal password prompt from the client to the user. This is useful when accessing a service that does not require a password.

Unless a password is specified on the command line or this parameter is specified, the client will request a password.

**-k** Try to authenticate with kerberos. Only useful in an Active Directory environment.

**-A|--authentication-file=filename**

This option allows you to specify a file from which to read the username and password used in the connection. The format of the file is

username = <value>

password = <value>

domain = <value>

Make certain that the permissions on the file restrict access from unwanted users.

**-U|--user=username[%password]**

Sets the SMB username or username and password.

If %password is not specified, the user will be prompted. The client will first check the **USER** environment variable, then the **LOGNAME** variable and if either exists, the string is uppercased. If these environmental variables are not found, the username **GUEST** is used.

A third option is to use a credentials file which contains the plaintext of the username and password. This option is mainly provided for scripts where the admin does not wish to pass the credentials on the command line or via environment variables. If this method is used, make certain that the permissions on the file restrict access from unwanted users. See the **-A** for more details.

Be cautious about including passwords in scripts. Also, on many systems the command line of a running process may be seen via the **ps** command. To be safe always allow **rpcclient** to prompt for a password and type it in directly.

**-n <primary NetBIOS name>**

This option allows you to override the NetBIOS name that Samba uses for itself. This is identical to setting the

parameter in the *smb.conf* file. However, a command line setting will take precedence over settings in *smb.conf*.

**-i <scope>**

This specifies a NetBIOS scope that **nmblookup** will use to communicate with when generating NetBIOS names. For details on the use of NetBIOS scopes, see rfc1001.txt and rfc1002.txt. NetBIOS scopes are **very** rarely used, only set this parameter if you are the system administrator in charge of all the NetBIOS systems you communicate with.

**-W|--workgroup=domain**

Set the SMB domain of the username. This overrides the default domain which is the domain defined in `smb.conf`. If the domain specified is the same as the servers NetBIOS name, it causes the client to log on using the servers local SAM (as opposed to the Domain SAM).

**-O socket options**

TCP socket options to set on the client socket. See the socket options parameter in the `smb.conf` manual page for the list of valid options.

**-T tar options**

`smbclient` may be used to create **tar(1)** compatible backups of all the files on an SMB/CIFS share. The secondary tar flags that can be given to this option are :

&#8226;

*c* - Create a tar file on UNIX. Must be followed by the name of a tar file, tape device or "-" for standard output. If using standard output you must turn the log level to its lowest value -d0 to avoid corrupting your tar file. This flag is mutually exclusive with the *x* flag.

&#8226;

*x* - Extract (restore) a local tar file back to a share. Unless the -D option is given, the tar files will be restored from the top level of the share. Must be followed by the name of the tar file, device or "-" for standard input. Mutually exclusive with the *c* flag. Restored files have their creation times (mtime) set to the date saved in the tar file. Directories currently do not get their creation dates restored properly.

&#8226;

*I* - Include files and directories. Is the default behavior when filenames are specified above. Causes files to be included in an extract or create (and therefore everything else to be excluded). See example below. Filename globbing works in one of two ways. See *r* below.

&#8226;

*X* - Exclude files and directories. Causes files to be excluded from an extract or create. See example below. Filename globbing works in one of two ways now. See *r* below.

&#8226;

*F* - File containing a list of files and directories. The *F* causes the name following the tarfile to create to be read as a filename that contains a list of files and directories to be included in an extract or create (and therefore everything else to be excluded). See example below. Filename globbing works in one of two ways. See *r* below.

&#8226;

*b* - Blocksize. Must be followed by a valid (greater than zero) blocksize. Causes tar file to be written out in blocksize\*TBLOCK (usually 512 byte) blocks.

&#8226;

*g* - Incremental. Only back up files that have the archive bit set. Useful only with the *c* flag.

&#8226;

*q* - Quiet. Keeps tar from printing diagnostics as it works. This is the same as `tarmode quiet`.

&#8226;

*r* - Regular expression include or exclude. Uses regular expression matching for excluding or excluding files if compiled with `HAVE_REGEX_H`. However this mode can be very slow. If not compiled with `HAVE_REGEX_H`, does a limited wildcard match on '\*' and '?'.  
&#8226;

&#8226;

*N* - Newer than. Must be followed by the name of a file whose date is compared against files found on the share during a create. Only files newer than the file specified are backed up to the tar file. Useful only with the *c* flag.

&#8226;

*a* - Set archive bit. Causes the archive bit to be reset when a file is backed up. Useful with the *g* and *c* flags.

### Tar Long File Names

**smbclient**'s tar option now supports long file names both on backup and restore. However, the full path name of the file must be less than 1024 bytes. Also, when a tar archive is created, **smbclient**'s tar option places all files in the archive with relative names, not absolute names.

### Tar Filenames

All file names can be given as DOS path names (with '\ ' as the component separator) or as UNIX path names (with '/' as the component separator).

### Examples

Restore from tar file *backup.tar* into myshare on mypc (no password on share).

```
smbclient //mypc/yshare "" -N -Tx backup.tar
```

Restore everything except *users/docs*

```
smbclient //mypc/myshare "" -N -TXx backup.tar users/docs
```

Create a tar file of the files beneath *users/docs*.

```
smbclient //mypc/myshare "" -N -Tc backup.tar users/docs
```

Create the same tar file as above, but now use a DOS path name.

```
smbclient //mypc/myshare "" -N -tc backup.tar users\docs
```

Create a tar file of the files listed in the file *tarlist*.

```
smbclient //mypc/myshare "" -N -TcF backup.tar tarlist
```

Create a tar file of all the files and directories in the share.

```
smbclient //mypc/myshare "" -N -Tc backup.tar *
```

**-D** initial directory

Change to initial directory before starting. Probably only of any use with the tar **-T** option.

**-c** command string

command string is a semicolon-separated list of commands to be executed instead of prompting from stdin.

**-N** is implied by **-c**.

This is particularly useful in scripts and for printing stdin to the server, e.g. **-c 'print -'**.

## OPERATIONS

Once the client is running, the user is presented with a prompt :

```
smb:>
```

The backslash ("\") indicates the current working directory on the server, and will change if the current

working directory is changed.

The prompt indicates that the client is ready and waiting to carry out a user command. Each command is a single word, optionally followed by parameters specific to that command. Command and parameters are space-delimited unless these notes specifically state otherwise. All commands are case-insensitive. Parameters to commands may or may not be case sensitive, depending on the command.

You can specify file names which have spaces in them by quoting the name with double quotes, for example "a long file name".

Parameters shown in square brackets (e.g., "[parameter]") are optional. If not given, the command will use suitable defaults. Parameters shown in angle brackets (e.g., "<parameter>") are required.

Note that all commands operating on the server are actually performed by issuing a request to the server. Thus the behavior may vary from server to server, depending on how the server was implemented.

The commands available are given here in alphabetical order.

? [command]

If *command* is specified, the ? command will display a brief informative message about the specified command. If no command is specified, a list of available commands will be displayed.

! [shell command]

If *shell command* is specified, the ! command will execute a shell locally and run the specified shell command. If no command is specified, a local shell will be run.

altname file

The client will request that the server return the "alternate" name (the 8.3 name) for a file or directory.

case\_sensitive

Toggles the setting of the flag in SMB packets that tells the server to treat filenames as case sensitive. Set to OFF by default (tells file server to treat filenames as case insensitive). Only currently affects Samba 3.0.5 and above file servers with the case sensitive parameter set to auto in the smb.conf.

cancel jobid0 [jobid1] ... [jobidN]

The client will request that the server cancel the printjobs identified by the given numeric print job ids.

chmod file mode in octal

This command depends on the server supporting the CIFS UNIX extensions and will fail if the server does not. The client requests that the server change the UNIX permissions to the given octal mode, in standard UNIX format.

chown file uid gid

This command depends on the server supporting the CIFS UNIX extensions and will fail if the server does not. The client requests that the server change the UNIX user and group ownership to the given decimal values. Note there is currently no way to remotely look up the UNIX uid and gid values for a given name. This may be addressed in future versions of the CIFS UNIX extensions.

cd [directory name]

If "directory name" is specified, the current working directory on the server will be changed to the directory specified. This operation will fail if for any reason the specified directory is inaccessible.

If no directory name is specified, the current working directory on the server will be reported.

del <mask>

The client will request that the server attempt to delete all files matching *mask* from the current working directory on the server.

dir <mask>

A list of the files matching *mask* in the current working directory on the server will be retrieved from the server and displayed.

**exit**

Terminate the connection with the server and exit from the program.

**get** <remote file name> [local file name]

Copy the file called *remote file name* from the server to the machine running the client. If specified, name the local copy *local file name*. Note that all transfers in **smbclient** are binary. See also the lowercase command.

**help** [command]

See the ? command above.

**lcd** [directory name]

If *directory name* is specified, the current working directory on the local machine will be changed to the directory specified. This operation will fail if for any reason the specified directory is inaccessible.

If no directory name is specified, the name of the current working directory on the local machine will be reported.

**link** target linkname

This command depends on the server supporting the CIFS UNIX extensions and will fail if the server does not. The client requests that the server create a hard link between the linkname and target files. The linkname file must not exist.

**lowercase**

Toggle lowercasing of filenames for the get and mget commands.

When lowercasing is toggled ON, local filenames are converted to lowercase when using the get and mget commands. This is often useful when copying (say) MSDOS files from a server, because lowercase filenames are the norm on UNIX systems.

**ls** <mask>

See the dir command above.

**mask** <mask>

This command allows the user to set up a mask which will be used during recursive operation of the mget and mput commands.

The masks specified to the mget and mput commands act as filters for directories rather than files when recursion is toggled ON.

The mask specified with the mask command is necessary to filter files within those directories. For example, if the mask specified in an mget command is "source\*" and the mask specified with the mask command is "\*.c" and recursion is toggled ON, the mget command will retrieve all files matching "\*.c" in all directories below and including all directories matching "source\*" in the current working directory.

Note that the value for mask defaults to blank (equivalent to "") and remains so until the mask command is used to change it. It retains the most recently specified value indefinitely. To avoid unexpected results it would be wise to change the value of mask back to "" after using the mget or mput commands.

**md** <directory name>

See the mkdir command.

**mget** <mask>

Copy all files matching *mask* from the server to the machine running the client.

Note that *mask* is interpreted differently during recursive operation and non-recursive operation - refer to the recurse and mask commands for more information. Note that all transfers in **smbclient** are binary. See also the lowercase command.

**mkdir** <directory name>

Create a new directory on the server (user access privileges permitting) with the specified name.

**mput** <mask>

Copy all files matching *mask* in the current working directory on the local machine to the current working directory on the server.

Note that *mask* is interpreted differently during recursive operation and non-recursive operation - refer to the **recurse** and **mask** commands for more information. Note that all transfers in **smbclient** are binary.

**print** <file name>

Print the specified file from the local machine through a printable service on the server.

**prompt**

Toggle prompting for filenames during operation of the **mget** and **mput** commands.

When toggled ON, the user will be prompted to confirm the transfer of each file during these commands. When toggled OFF, all specified files will be transferred without prompting.

**put** <local file name> [remote file name]

Copy the file called *local file name* from the machine running the client to the server. If specified, name the remote copy *remote file name*. Note that all transfers in **smbclient** are binary. See also the lowercase command.

**queue**

Displays the print queue, showing the job id, name, size and current status.

**quit**

See the exit command.

**rd** <directory name>

See the **rmdir** command.

**recurse**

Toggle directory recursion for the commands **mget** and **mput**.

When toggled ON, these commands will process all directories in the source directory (i.e., the directory they are copying from ) and will recurse into any that match the mask specified to the command. Only files that match the mask specified using the **mask** command will be retrieved. See also the **mask** command.

When recursion is toggled OFF, only files from the current working directory on the source machine that match the mask specified to the **mget** or **mput** commands will be copied, and any mask specified using the **mask** command will be ignored.

**rm** <mask>

Remove all files matching *mask* from the current working directory on the server.

**rmdir** <directory name>

Remove the specified directory (user access privileges permitting) from the server.

**setmode** <filename> <perm=[+|-]rsha>

A version of the DOS **attrib** command to set file permissions. For example:

**setmode myfile +r**

would make myfile read only.

**stat** file

This command depends on the server supporting the CIFS UNIX extensions and will fail if the server does not. The client requests the UNIX basic info level and prints out the same info that the Linux **stat**

command would about the file. This includes the size, blocks used on disk, file type, permissions, inode number, number of links and finally the three timestamps (access, modify and change). If the file is a special file (symlink, character or block device, fifo or socket) then extra information may also be printed.

symlink target linkname

This command depends on the server supporting the CIFS UNIX extensions and will fail if the server does not. The client requests that the server create a symbolic hard link between the target and linkname files. The linkname file must not exist. Note that the server will not create a link to any path that lies outside the currently connected share. This is enforced by the Samba server.

tar <c|x>[IXbgNa]

Performs a tar operation - see the *-T* command line option above. Behavior may be affected by the tar-mode command (see below). Using *g* (incremental) and *N* (newer) will affect tarmode settings. Note that using the *"-"* option with tar *x* may not work - use the command line option instead.

blocksize <blocksize>

Blocksize. Must be followed by a valid (greater than zero) blocksize. Causes tar file to be written out in *blocksize*\*TBLOCK (usually 512 byte) blocks.

tarmode <full|inc|reset|noreset>

Changes tar's behavior with regard to archive bits. In full mode, tar will back up everything regardless of the archive bit setting (this is the default mode). In incremental mode, tar will only back up files with the archive bit set. In reset mode, tar will reset the archive bit on all files it backs up (implies read/write share).

## NOTES

Some servers are fussy about the case of supplied usernames, passwords, share names (AKA service names) and machine names. If you fail to connect try giving all parameters in uppercase.

It is often necessary to use the *-n* option when connecting to some types of servers. For example OS/2 Lan-Manager insists on a valid NetBIOS name being used, so you need to supply a valid name that would be known to the server.

smbclient supports long file names where the server supports the LANMAN2 protocol or above.

## ENVIRONMENT VARIABLES

The variable **USER** may contain the username of the person using the client. This information is used only if the protocol level is high enough to support session-level passwords.

The variable **PASSWD** may contain the password of the person using the client. This information is used only if the protocol level is high enough to support session-level passwords.

The variable **LIBSMB\_PROG** may contain the path, executed with *system()*, which the client should connect to instead of connecting to a server. This functionality is primarily intended as a development aid, and works best when using a LMHOSTS file

## INSTALLATION

The location of the client program is a matter for individual system administrators. The following are thus suggestions only.

It is recommended that the smbclient software be installed in the */usr/local/samba/bin/* or */usr/samba/bin/* directory, this directory readable by all, writeable only by root. The client program itself should be executable by all. The client should **NOT** be setuid or setgid!

The client log files should be put in a directory readable and writeable only by the user.

To test the client, you will need to know the name of a running SMB/CIFS server. It is possible to run **smbd**(8) as an ordinary user - running that server as a daemon on a user-accessible port (typically any port number over 1024) would provide a suitable test server.

**DIAGNOSTICS**

Most diagnostics issued by the client are logged in a specified log file. The log file name is specified at compile time, but may be overridden on the command line.

The number and nature of diagnostics available depends on the debug level used by the client. If you have problems, set the debug level to 3 and peruse the log files.

**VERSION**

This man page is correct for version 3 of the Samba suite.

**AUTHOR**

The original Samba software and related utilities were created by Andrew Tridgell. Samba is now developed by the Samba Team as an Open Source project similar to the way the Linux kernel is developed.

The original Samba man pages were written by Karl Auer. The man page sources were converted to YODL format (another excellent piece of Open Source software, available at <ftp://ftp.icce.rug.nl/pub/unix/>) and updated for the Samba 2.0 release by Jeremy Allison. The conversion to DocBook for Samba 2.2 was done by Gerald Carter. The conversion to DocBook XML 4.2 for Samba 3.0 was done by Alexander Bokovoy.