Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'smb.conf.5' command

$ man smb.conf.5

SMB.CONF(5) File Formats and Conventions SMB.CONF(5)

NAME

smb.conf - The configuration file for the Samba suite

SYNOPSIS

The smb.conf file is a configuration file for the Samba suite.

smb.conf contains runtime configuration information for the Samba programs. The complete description of the file format and possible parameters held within are here for reference purposes.

HOW CONFIGURATION CHANGES ARE APPLIED

The Samba suite includes a number of different programs. Some of them operate in a client mode, others are server daemons that provide various services to its clients. The smb.conf file is processed in the following way:

? The Samba suite's client applications read their configuration only once. Any changes made after start aren't reflected in the context of already running client code.

? The Samba suite's server daemons reload their configuration when requested. However, already active connections do not change their configuration. More detailed information can be found in smbd(8) and winbindd(8) manual pages.

To request Samba server daemons to refresh their configuration, please use smbcontrol(1) utility.

FILE FORMAT

The file consists of sections and parameters. A section begins with the name of the section in square brackets and continues until the next section begins. Sections contain parameters of the form:

```
name = value
```

The file is line-based - that is, each newline-terminated line represents either a comment, a section name or a parameter.

Section and parameter names are not case sensitive.

Only the first equals sign in a parameter is significant. Whitespace before or after the first equals sign is discarded. Leading, trailing and internal whitespace in section and parameter names is irrelevant. Leading and trailing whitespace in a parameter value is discarded. Internal whitespace within a parameter value is retained verbatim.

Any line beginning with a semicolon (?;?) or a hash (?#?) character is ignored, as are lines containing only whitespace.

Any line ending in a ?\? is continued on the next line in the customary UNIX fashion.

The values following the equals sign in parameters are all either a string (no quotes needed) or a boolean, which may be given as yes/no, 1/0 or true/false. Case is not significant in boolean values, but is preserved in string values. Some items such as create masks are numeric.

SECTION DESCRIPTIONS

Each section in the configuration file (except for the [global] section) describes a shared resource (known as a ?share?). The section name is the name of the shared resource and the parameters within the section define the shares attributes.

There are three special sections, [global], [homes] and [printers], which are described under special sections. The following notes apply to ordinary section descriptions.

A share consists of a directory to which access is being given plus a description of the access rights which are granted to the user of the service. Some housekeeping options are also specifiable.

Sections are either file share services (used by the client as an extension of their native file systems) or printable services (used by the client to access print services on the host running the server).

Sections may be designated guest services, in which case no password is required to access them. A specified UNIX guest account is used to define access privileges in this case.

Sections other than guest services will require a password to access them. The client provides the username. As older clients only provide passwords and not usernames, you may specify a list of usernames to check against the password using the user = option in the share definition. For modern clients such as Windows 95/98/ME/NT/2000, this should not be necessary.

The access rights granted by the server are masked by the access rights granted to the specified or guest UNIX user by the host system. The server does not grant more access than the host system grants.

The following sample section defines a file space share. The user has write access to the path /home/bar. The share is accessed via the share name foo:

```
[foo]
path = /home/bar
read only = no
```

The following sample section defines a printable share. The share is read-only, but printable. That is, the only write access permitted is via calls to open, write to and close a spool file. The guest ok parameter means access will be permitted as the default guest user (specified elsewhere):

```
[aprinter]
path = /var/tmp
read only = yes
printable = yes
guest ok = yes
```

SPECIAL SECTIONS

The [global] section

Parameters in this section apply to the server as a whole, or are defaults for sections that do not specifically define certain items.

See the notes under PARAMETERS for more information.

The [homes] section

If a section called [homes] is included in the configuration file, services connecting clients to their home directories can be created on the fly by the server.

When the connection request is made, the existing sections are scanned. If a match is found, it is used. If no match is found, the requested section name is treated as a username and looked up in the local password file. If the name exists and the correct password has been given, a share is created by cloning the [homes] section.

Some modifications are then made to the newly created share:

? The share name is changed from homes to the located username.

? If no path was given, the path is set to the user's home directory.

If you decide to use a path = line in your [homes] section, it may be useful to use the %S macro. For example:

```
path = /data/pchome/%S
```

is useful if you have different home directories for your PCs than for UNIX access.

This is a fast and simple way to give a large number of clients access to their home directories with a minimum of fuss.

A similar process occurs if the requested section name is ?homes?, except that the share name is not changed to that of the requesting user. This method of using the [homes] section works well if different users share a client PC.

The [homes] section can specify all the parameters a normal service section can specify, though some make more sense than others. The following is a typical and suitable [homes] section:

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

An important point is that if guest access is specified in the [homes] section, all home directories will be visible to all clients without a password. In the very unlikely event that this is actually desirable, it is wise to also specify read only access.

The browseable flag for auto home directories will be inherited from the global browseable flag, not the [homes] browseable flag. This is useful as it means setting browseable = no in the [homes] section will hide the [homes] share but make any auto home directories visible.

The [printers] section

This section works like [homes], but for printers.

If a [printers] section occurs in the configuration file, users are able to connect to any printer specified in the local host's printcap file.

When a connection request is made, the existing sections are scanned. If a match is found, it is used. If no match is found, but a [homes] section exists, it is used as described above. Otherwise, the requested section name is treated as a printer name and the appropriate printcap file is scanned to see if the requested section name is a valid printer share name. If a match is found, a new printer share is created by cloning the [printers] section.

A few modifications are then made to the newly created share:

? The share name is set to the located printer name

? If no printer name was given, the printer name is set to the located printer name

? If the share does not permit guest access and no username was given, the username is set to the located printer name.

The [printers] service MUST be printable - if you specify otherwise, the server will refuse to load the configuration file.

Typically the path specified is that of a world-writeable spool directory with the sticky bit set on it. A typical [printers] entry looks like this:

```
[printers]
path = /var/tmp
guest ok = yes
```

```
printable = yes
```

All aliases given for a printer in the printcap file are legitimate printer names as far as the server is concerned. If your printing subsystem doesn't work like that, you will have to set up a pseudo-printcap. This is a file consisting of one or more lines like this:

```
alias|alias|alias|alias...
```

Each alias should be an acceptable printer name for your printing subsystem. In the [global] section, specify the new file as your printcap. The server will only recognize names found in your pseudo-printcap, which of course can contain whatever aliases you like. The same technique could be used simply to limit access to a subset of your local printers.

An alias, by the way, is defined as any component of the first entry of a printcap record. Records are separated by newlines, components (if there are more than one) are separated by vertical bar symbols (|).

> Note
>
> On SYSV systems which use lpstat to determine what printers are defined on the system you may be able to use printcap name = lpstat to automatically obtain a list of printers. See the printcap name option for more details.

USERSHARES

Starting with Samba version 3.0.23 the capability for non-root users to add, modify, and delete their own share definitions has been added. This capability is called usershares and is controlled by a set of parameters in the [global] section of the smb.conf. The relevant parameters are :

usershare allow guests

Controls if usershares can permit guest access.

usershare max shares

Maximum number of user defined shares allowed.

usershare owner only

If set only directories owned by the sharing user can be shared.

usershare path

Points to the directory containing the user defined share definitions. The filesystem permissions on this directory control who can create user defined shares.

usershare prefix allow list

Comma-separated list of absolute pathnames restricting what directories can be shared. Only directories below the pathnames in this list are permitted.

usershare prefix deny list

Comma-separated list of absolute pathnames restricting what directories can be shared. Directories below the pathnames in this list are prohibited.

usershare template share

Names a pre-existing share used as a template for creating new usershares. All other share parameters not specified in the user defined share definition are copied from this named share.

To allow members of the UNIX group foo to create user defined shares, create the directory to contain the share definitions as follows:

Become root:

```
mkdir /usr/local/samba/lib/usershares
chgrp foo /usr/local/samba/lib/usershares
chmod 1770 /usr/local/samba/lib/usershares
```

Then add the parameters

```
    usershare path = /usr/local/samba/lib/usershares
    usershare max shares = 10 # (or the desired number of shares)
```

to the global section of your smb.conf. Members of the group foo may then manipulate the user defined shares using the following commands.

net usershare add sharename path [comment] [acl] [guest_ok=[y|n]]

To create or modify (overwrite) a user defined share.

net usershare delete sharename

To delete a user defined share.

net usershare list wildcard-sharename

To list user defined shares.

net usershare info wildcard-sharename

To print information about user defined shares.

PARAMETERS

Parameters define the specific attributes of sections.

Some parameters are specific to the [global] section (e.g., security).

Some parameters are usable in all sections (e.g., create mask). All others are permissible only in normal sections. For the purposes of the following descriptions the [homes] and [printers] sections will be considered normal. The letter G in parentheses indicates that a parameter is specific to the [global] section. The letter S indicates that a parameter can be specified in a service specific section. All S parameters can also be specified in the [global] section - in which case they will define the default behavior for all services.

Parameters are arranged here in alphabetical order - this may not create best bedfellows, but at least you can find them! Where there are synonyms, the preferred synonym is described, others refer to the preferred synonym.

VARIABLE SUBSTITUTIONS

Many of the strings that are settable in the config file can take substitutions. For example the option ?path = /tmp/%u? is interpreted as ?path = /tmp/john? if the user connected with the username john. These substitutions are mostly noted in the descriptions below, but there are some general substitutions which apply whenever they might be relevant. These are:

%U

session username (the username that the client wanted, not necessarily the same as the one they got).

%G

primary group name of %U.

%h

the Internet hostname that Samba is running on.

%m

the NetBIOS name of the client machine (very useful).

This parameter is not available when Samba listens on port 445, as clients no longer send this information. If you use this macro in an include statement on a domain that has a Samba domain controller be sure to set in the [global] section smb ports = 139. This will cause Samba to not listen on port 445 and will permit include functionality to function as it did with Samba 2.x.

%L

the NetBIOS name of the server. This allows you to change your config based on what the client calls you. Your server can have a ?dual personality?.

%M

the Internet name of the client machine.

%R

the selected protocol level after protocol negotiation. It can be one of CORE, COREPLUS, LANMAN1, LANMAN2, NT1, SMB2_02, SMB2_10, SMB3_00, SMB3_02, SMB3_11 or SMB2_FF.

%d

the process id of the current server process.

%a

The architecture of the remote machine. It currently recognizes Samba (Samba), the Linux CIFS file system (CIFSFS), OS/2, (OS2), Mac OS X (OSX), Windows for Workgroups (WfWg), Windows 9x/ME (Win95), Windows NT (WinNT), Windows 2000 (Win2K), Windows XP (WinXP), Windows XP 64-bit(WinXP64), Windows 2003 including 2003R2 (Win2K3), and Windows Vista (Vista). Anything else will be known as UNKNOWN.

%I

the IP address of the client machine.

Before 4.0.0 it could contain IPv4 mapped IPv6 addresses, now it only contains IPv4 or IPv6 addresses.

%J

the IP address of the client machine, colons/dots replaced by underscores.

%i

the local IP address to which a client connected.

Before 4.0.0 it could contain IPv4 mapped IPv6 addresses, now it only contains IPv4 or IPv6 addresses.

%j

the local IP address to which a client connected, colons/dots replaced by underscores.

%T

the current date and time.

%t

the current date and time in a minimal format without colons (YYYYYmmdd_HHMMSS).

%D

name of the domain or workgroup of the current user.

%w

the winbind separator.

%$(envvar)

the value of the environment variable envar.

The following substitutes apply only to some configuration options (only those that are used when a connection has been established):

%S

the name of the current service, if any.

%P

the root directory of the current service, if any.

%u

username of the current service, if any.

%g

primary group name of %u.

%H

the home directory of the user given by %u.

%N

This value is the same as %L.

There are some quite creative things that can be done with these substitutions and other smb.conf options.

NAME MANGLING

Samba supports name mangling so that DOS and Windows clients can use files that don't conform to the 8.3 format. It can also be set to adjust the case of 8.3 format filenames.

There are several options that control the way mangling is performed, and they are grouped here rather than listed separately. For the defaults look at the output of the testparm program.

These options can be set separately for each service.

The options are:

case sensitive = yes/no/auto

controls whether filenames are case sensitive. If they aren't, Samba must do a filename search and match on passed names. The default setting of auto allows clients that support case sensitive filenames (Linux CIFSVFS and smbclient 3.0.5 and above currently) to tell the Samba server on a per-packet basis that they wish to access the file system in a case-sensitive manner (to support UNIX case sensitive semantics). No Windows or DOS system supports case-sensitive filename so setting this option to auto is that same as setting it to no for them. Default auto.

default case = upper/lower

controls what the default case is for new filenames (ie. files that don't currently exist in the filesystem). Default lower. IMPORTANT NOTE: As part of the optimizations for directories containing large numbers of files, the following special case applies. If the options case sensitive = yes, preserve case = No, and short preserve case = No are set, then the case of all incoming client filenames, not just new filenames, will be modified. See additional notes below.

preserve case = yes/no

controls whether new files (ie. files that don't currently exist in the filesystem) are created with the case that the client passes, or if they are forced to be the default case. Default yes.

short preserve case = yes/no

controls if new files (ie. files that don't currently exist in the filesystem) which conform to 8.3 syntax, that is all in upper case and of suitable length, are created upper case, or if they are forced to be the default case. This option can be used with preserve case = yes to permit long filenames to retain their case, while short names are lowercased. Default yes.

By default, Samba 3.0 has the same semantics as a Windows NT server, in that it is case insensitive but case preserving. As a special case for directories with large numbers of files, if the case options are set as follows, "case sensitive = yes", "case preserve = no", "short preserve case = no" then the "default case" option will be applied and will modify all filenames sent from the client when accessing this share.

REGISTRY-BASED CONFIGURATION

Starting with Samba version 3.2.0, the capability to store Samba configuration in the registry is available. The configuration is stored in the registry key HKLM\Software\Samba\smbconf. There are two levels of registry configuration:

1. Share definitions stored in registry are used. This is triggered by setting the global parameter registry shares to ?yes? in smb.conf.

   The registry shares are loaded not at startup but on demand at runtime by smbd. Shares defined in smb.conf take priority over shares of the same name defined in registry.

2. Global smb.conf options stored in registry are used. This can be activated in two different ways:

   Firstly, a registry only configuration is triggered by setting config backend = registry in the [global] section of smb.conf. This resets everything that has been read from config files to this point and reads the content of the global configuration section from the registry. This is the recommended method of using registry based configuration.

   Secondly, a mixed configuration can be activated by a special new meaning of the parameter include = registry in the [global] section of smb.conf. This reads the global options from registry with the same priorities as for an include of a text file. This may be especially useful in cases where an initial configuration is needed to access the registry.

Activation of global registry options automatically activates registry shares. So in the registry only case, shares are loaded on demand only.

Note: To make registry-based configurations foolproof at least to a certain extent, the use of lock directory and config backend inside the registry configuration has been disabled: Especially by changing the lock directory inside the registry configuration, one would create a broken setup where the daemons do not see the configuration they loaded once it is active.

The registry configuration can be accessed with tools like regedit or net (rpc) registry in the key HKLM\Software\Samba\smbconf. More conveniently, the conf subcommand of the net(8) utility offers a dedicated interface to read and write the registry based configuration locally, i.e. directly accessing the database file, circumventing the server.

IDENTITY MAPPING CONSIDERATIONS

In the SMB protocol, users, groups, and machines are represented by their security identifiers (SIDs). On POSIX system Samba processes need to run under corresponding POSIX user identities and with supplemental POSIX groups to allow access to the files owned by those users and groups. The process of mapping SIDs to POSIX users and groups is called IDENTITY MAPPING or, in short, ID MAPPING.

Samba supports multiple ways to map SIDs to POSIX users and groups. The configuration is driven by the idmap config DOMAIN : OPTION option which allows one to specify identity mapping (idmap) options for each domain separately.

Identity mapping modules implement different strategies for mapping of

SIDs to POSIX user and group identities. They are applicable to different use cases and scenarios. It is advised to read the documentation of the individual identity mapping modules before choosing a specific scenario to use. Each identity management module is documented in a separate manual page. The standard idmap backends are tdb (idmap_tdb(8)), tdb2 (idmap_tdb2(8)), ldap (idmap_ldap(8)), rid (idmap_rid(8)), hash (idmap_hash(8)), autorid (idmap_autorid(8)), ad (idmap_ad(8)), nss (idmap_nss(8)), and rfc2307 (idmap_rfc2307(8)).

Overall, ID mapping configuration should be decided carefully. Changes to the already deployed ID mapping configuration may create the risk of losing access to the data or disclosing the data to the wrong parties.

This example shows how to configure two domains with idmap_rid(8), the principal domain and a trusted domain, leaving the default id mapping scheme at tdb.

```
[global]
security = domain
workgroup = MAIN
idmap config * : backend        = tdb
idmap config * : range          = 1000000-1999999
idmap config MAIN : backend     = rid
idmap config MAIN : range       = 5000000-5999999
idmap config TRUSTED : backend  = rid
idmap config TRUSTED : range    = 6000000-6999999
```

EXPLANATION OF EACH PARAMETER

abort shutdown script (G)

This a full path name to a script called by smbd(8) that should stop a shutdown procedure issued by the shutdown script.

If the connected user possesses the SeRemoteShutdownPrivilege, right, this command will be run as root.

Default: abort shutdown script = ""

Example: abort shutdown script = /sbin/shutdown -c

access based share enum (S)

If this parameter is yes for a service, then the share hosted by the service will only be visible to users who have read or write access to the share during share enumeration (for example net view \\sambaserver). The share ACLs which allow or deny the access to the share can be modified using for example the sharesec command or using the appropriate Windows tools. This has parallels to access based enumeration, the main difference being that only share permissions are evaluated, and security descriptors on files contained on the share are not used in computing enumeration access rights.

Default: access based share enum = no

acl allow execute always (S)

This boolean parameter controls the behaviour of smbd(8) when receiving a protocol request of "open for execution" from a Windows client. With Samba 3.6 and older, the execution right in the ACL was not checked, so a client could execute a file even if it did not have execute rights on the file. In Samba 4.0, this has been fixed, so that by default, i.e. when this parameter is set to "False", "open for execution" is now denied when execution permissions are not present.

If this parameter is set to "True", Samba does not check execute permissions on "open for execution", thus re-establishing the behaviour of Samba 3.6. This can be useful to smoothen upgrades from older Samba versions to 4.0 and newer. This setting is not meant to be used as a permanent setting, but as a temporary relief: It is recommended to fix the permissions in the ACLs and reset this parameter to the default after a certain transition period.

Default: acl allow execute always = no

acl check permissions (S)

Please note this parameter is now deprecated in Samba 3.6.2 and will be removed in a future version of Samba.

This boolean parameter controls what smbd(8) does on receiving a protocol request of "open for delete" from a Windows client. If a Windows client doesn't have permissions to delete a file then they expect this to be denied at open time. POSIX systems normally only detect restrictions on delete by actually attempting to delete the file or directory. As Windows clients can (and do) "back out" a delete request by unsetting the "delete on close" bit Samba cannot delete the file immediately on "open for delete" request as we cannot restore such a deleted file. With this parameter set to true (the default) then smbd checks the file system permissions directly on "open for delete" and denies the request without actually deleting the file if the file system permissions would seem to deny it. This is not perfect, as it's possible a user could have deleted a file without Samba being able to check the permissions correctly, but it is close enough to Windows semantics for mostly correct behaviour. Samba will correctly check POSIX ACL semantics in this case.

If this parameter is set to "false" Samba doesn't check permissions on "open for delete" and allows the open. If the user doesn't have permission to delete the file this will only be discovered at close time, which is too late for the Windows user tools to display an error message to the user. The symptom of this is files that appear to have been deleted "magically" re-appearing on a Windows explorer refresh. This is an extremely advanced protocol option which should not need to be changed. This parameter was introduced in its final form in 3.0.21, an earlier version with slightly different semantics was introduced in 3.0.20. That older version is not documented here.

Default: acl check permissions = yes

acl flag inherited canonicalization (S)

This option controls the way Samba handles client requests setting the Security Descriptor of files and directories and the effect the operation has on the Security Descriptor flag "DACL auto-inherited" (DI). Generally, this flag is set on a file (or directory) upon creation if the parent directory has DI set and also has inheritable ACEs.

On the other hand when a Security Descriptor is explicitly set on a file, the DI flag is cleared, unless the flag "DACL Inheritance Required" (DR) is also set in the new Security Descriptor (fwiw, DR is never stored on disk).

This is the default behaviour when this option is enabled (the default). When setting this option to no, the resulting value of the DI flag on-disk is directly taken from the DI value of the to-be-set Security Descriptor. This can be used so dump tools like rsync that copy data blobs from xattrs that represent ACLs created by the acl_xattr VFS module will result in copies of the ACL that are identical to the source. Without this option, the copied ACLs would all loose the DI flag if set on the source.

Default: acl flag inherited canonicalization = yes

acl group control (S)

In a POSIX filesystem, only the owner of a file or directory and the superuser can modify the permissions and ACLs on a file. If this parameter is set, then Samba overrides this restriction, and also allows the primary group owner of a file or directory to modify the permissions and ACLs on that file.

On a Windows server, groups may be the owner of a file or directory - thus allowing anyone in that group to modify the permissions on it. This allows the delegation of security controls on a point in the filesystem to the group owner of a directory and anything below it also owned by that group. This means there are multiple people with permissions to modify ACLs on a file or directory, easing manageability.

This parameter allows Samba to also permit delegation of the control over a point in the exported directory hierarchy in much the same way as Windows. This allows all members of a UNIX group to control the permissions on a file or directory they have group ownership on.

This parameter is best used with the inherit owner option and also on a share containing directories with the UNIX setgid bit set on them, which causes new files and directories created within it to inherit the group ownership from the containing directory.

This parameter was deprecated in Samba 3.0.23, but re-activated in Samba 3.0.31 and above, as it now only controls permission changes if the user is in the owning primary group. It is now no longer equivalent to the dos filemode option.

Default: acl group control = no

acl map full control (S)

This boolean parameter controls whether smbd(8) maps a POSIX ACE entry of "rwx" (read/write/execute), the maximum allowed POSIX permission set, into a Windows ACL of "FULL CONTROL". If this parameter is set to true any POSIX ACE entry of "rwx" will be returned in a Windows ACL as "FULL CONTROL", is this parameter is set to false any POSIX ACE entry of "rwx" will be returned as the specific Windows ACL bits representing read, write and execute.

Default: acl map full control = yes

add group script (G)

This is the full pathname to a script that will be run AS ROOT by smbd(8) when a new group is requested. It will expand any %g to the group name passed. This script is only useful for installations using the Windows NT domain administration tools. The script is free to create a group with an arbitrary name to circumvent unix group name restrictions. In that case the script must print the numeric gid of the created group on stdout.

Default: add group script =

Example: add group script = /usr/sbin/groupadd %g

additional dns hostnames (G)

A list of additional DNS names by which this host can be identified

Default: additional dns hostnames = # empty string (no additional dns names)

Example: additional dns hostnames = host2.example.com host3.other.com

add machine script (G)

This is the full pathname to a script that will be run by smbd(8) when a machine is added to Samba's domain and a Unix account matching the machine's name appended with a "$" does not already exist.

This option is very similar to the add user script, and likewise uses the %u substitution for the account name. Do not use the %m substitution.

Default: add machine script =

Example: add machine script = /usr/sbin/adduser -n -g machines -c Machine -d /var/lib/nobody -s /bin/false %u

addport command (G)

Samba 3.0.23 introduced support for adding printer ports remotely using the Windows "Add Standard TCP/IP Port Wizard". This option defines an external program to be executed when smbd receives a request to add a new Port to the system. The script is passed two parameters:

? port name

? device URI

The deviceURI is in the format of socket://<hostname>[:<portnumber>] or lpd://<hostname>/<queuename>.

Default: addport command =

Example: addport command = /etc/samba/scripts/addport.sh

addprinter command (G)

With the introduction of MS-RPC based printing support for Windows NT/2000 clients in Samba 2.2, The MS Add Printer Wizard (APW) icon is now also available in the "Printers..." folder displayed a share listing. The APW allows for printers to be add remotely to a Samba or Windows NT/2000 print server.

For a Samba host this means that the printer must be physically added to the underlying printing system. The addprinter command defines a script to be run which will perform the necessary operations for adding the printer to the print system and to add the appropriate service definition to the smb.conf file in order that it can be shared by smbd(8).

The addprinter command is automatically invoked with the following parameter (in order):

? printer name

? share name

? port name

? driver name

? location

? Windows 9x driver location

All parameters are filled in from the PRINTER_INFO_2 structure sent by the Windows NT/2000 client with one exception. The "Windows 9x driver location" parameter is included for backwards compatibility only. The remaining fields in the structure are generated from answers to the APW questions.

Once the addprinter command has been executed, smbd will reparse the smb.conf to determine if the share defined by the APW exists. If the sharename is still invalid, then smbd will return an ACCESS_DENIED error to the client.

The addprinter command program can output a single line of text, which Samba will set as the port the new printer is connected to. If this line isn't output, Samba won't reload its printer shares.

Default: addprinter command =

Example: addprinter command = /usr/bin/addprinter

add share command (G)

Samba 2.2.0 introduced the ability to dynamically add and delete shares via the Windows NT 4.0 Server Manager. The add share command is used to define an external program or script which will add a new service definition to smb.conf.

In order to successfully execute the add share command, smbd requires that the administrator connects using a root account (i.e. uid == 0) or has the SeDiskOperatorPrivilege. Scripts defined in the add share command parameter are executed as root.

When executed, smbd will automatically invoke the add share command with five parameters.

? configFile - the location of the global smb.conf file.

? shareName - the name of the new share.

? pathName - path to an **existing** directory on disk.

? comment - comment string to associate with the new share.

? max connections Number of maximum simultaneous connections to this share.

This parameter is only used to add file shares. To add printer shares, see the addprinter command.

Default: add share command =

Example: add share command = /usr/local/bin/addshare

add user script (G)

This is the full pathname to a script that will be run AS ROOT by smbd(8) under special circumstances described below.

Normally, a Samba server requires that UNIX users are created for all users accessing files on this server. For sites that use Windows NT account databases as their primary user database creating these users and keeping the user list in sync with the Windows NT PDC is an onerous task. This option allows smbd to create the required UNIX users ON DEMAND when a user accesses the Samba server.

When the Windows user attempts to access the Samba server, at login (session setup in the SMB protocol) time, smbd(8) contacts the password server and attempts to authenticate the given user with the given password. If the authentication succeeds then smbd attempts to find a UNIX user in the UNIX password database to map the Windows user into. If this lookup fails, and add user script is set then smbd will call the specified script AS ROOT, expanding any %u argument to be the user name to create.

If this script successfully creates the user then smbd will continue on as though the UNIX user already existed. In this way, UNIX users are dynamically created to match existing Windows NT accounts.

See also security, password server, delete user script.

Default: add user script =

Example: add user script = /usr/local/samba/bin/add_user %u

add user to group script (G)

Full path to the script that will be called when a user is added to a group using the Windows NT domain administration tools. It will be run by smbd(8) AS ROOT. Any %g will be replaced with the group name and any %u will be replaced with the user name.

Note that the adduser command used in the example below does not support the used syntax on all systems.

Default: add user to group script =

Example: add user to group script = /usr/sbin/adduser %u %g

administrative share (S)

If this parameter is set to yes for a share, then the share will be an administrative share. The Administrative Shares are the default network shares created by all Windows NT-based operating systems. These are shares like C$, D$ or ADMIN$. The type of these shares is STYPE_DISKTREE_HIDDEN.

See the section below on security for more information about this option.

Default: administrative share = no

admin users (S)

This is a list of users who will be granted administrative privileges on the share. This means that they will do all file operations as the super-user (root).

You should use this option very carefully, as any user in this list will be able to do anything they like on the share, irrespective of file permissions.

Default: admin users =

Example: admin users = jason

afs share (S)

This parameter controls whether special AFS features are enabled for this share. If enabled, it assumes that the directory exported via the path parameter is a local AFS import. The special AFS features include the attempt to hand-craft an AFS token if you enabled --with-fake-kaserver in configure.

Default: afs share = no

afs token lifetime (G)

This parameter controls the lifetime of tokens that the AFS fake-kaserver claims. In reality these never expire but this lifetime controls when the afs client will forget the token.

Set this parameter to 0 to get NEVERDATE.

Default: afs token lifetime = 604800

afs username map (G)

If you are using the fake kaserver AFS feature, you might want to hand-craft the usernames you are creating tokens for. For example this is necessary if you have users from several domain in your AFS Protection Database. One possible scheme to code users as DOMAIN+User as it is done by winbind with the + as a separator.

The mapped user name must contain the cell name to log into, so without setting this parameter there will be no token.

Default: afs username map =

Example: afs username map = %u@afs.samba.org

aio max threads (G)

The integer parameter specifies the maximum number of threads each smbd process will create when doing parallel asynchronous IO calls. If the number of outstanding calls is greater than this number the requests will not be refused but go onto a queue and will be scheduled in turn as outstanding requests complete.

Related command: aio read size

Related command: aio write size

Default: aio max threads = 100

aio read size (S)

If this integer parameter is set to a non-zero value, Samba will read from files asynchronously when the request size is bigger than this value. Note that it happens only for non-chained and non-chaining reads and when not using write cache.

The only reasonable values for this parameter are 0 (no async I/O) and 1 (always do async I/O).

Related command: write cache size

Related command: aio write size

Default: aio read size = 1

Example: aio read size = 0 # Always do reads synchronously

aio write behind (S)

If Samba has been built with asynchronous I/O support, Samba will not wait until write requests are finished before returning the result to the client for files listed in this parameter. Instead, Samba will immediately return that the write request has been finished successfully, no matter if the operation will succeed or not. This might speed up clients without aio support, but is really dangerous, because data could be lost and files could be damaged.

The syntax is identical to the veto files parameter.

Default: aio write behind =

Example: aio write behind = /*.tmp/

aio write size (S)

If this integer parameter is set to a non-zero value, Samba will write to files asynchronously when the request size is bigger than this value. Note that it happens only for non-chained and non-chaining reads and when not using write cache.

The only reasonable values for this parameter are 0 (no async I/O) and 1 (always do async I/O).

Compared to aio read size this parameter has a smaller effect, most writes should end up in the file system cache. Writes that require space allocation might benefit most from going asynchronous.

Related command: write cache size

Related command: aio read size

Default: aio write size = 1

Example: aio write size = 0 # Always do writes synchronously algorithmic rid base (G)

This determines how Samba will use its algorithmic mapping from uids/gid to the RIDs needed to construct NT Security Identifiers.

Setting this option to a larger value could be useful to sites transitioning from WinNT and Win2k, as existing user and group rids would otherwise clash with system users etc.

All UIDs and GIDs must be able to be resolved into SIDs for the correct operation of ACLs on the server. As such the algorithmic mapping can't be 'turned off', but pushing it 'out of the way' should resolve the issues. Users and groups can then be assigned 'low' RIDs in arbitrary-rid supporting backends.

Default: algorithmic rid base = 1000

Example: algorithmic rid base = 100000

allocation roundup size (S)

This parameter allows an administrator to tune the allocation size reported to Windows clients. This is only useful for old SMB1 clients because modern SMB dialects eliminated that bottleneck and have better performance by default. Using this parameter may cause difficulties for some applications, e.g. MS Visual Studio. If the MS Visual Studio compiler starts to crash with an internal error, set this parameter to zero for this share. Settings this parameter to a large value can also cause small files to allocate more space on the disk than needed.

This parameter is deprecated and will be removed in one of the next Samba releases.

The integer parameter specifies the roundup size in bytes.

Default: allocation roundup size = 0

Example: allocation roundup size = 1048576 # (to set it to the former default of 1 MiB)

allow dcerpc auth level connect (G)

This option controls whether DCERPC services are allowed to be used with DCERPC_AUTH_LEVEL_CONNECT, which provides authentication, but no per message integrity nor privacy protection.

Some interfaces like samr, lsarpc and netlogon have a hard-coded default of no and epmapper, mgmt and rpcecho have a hard-coded default of yes.

The behavior can be overwritten per interface name (e.g. lsarpc, netlogon, samr, srvsvc, winreg, wkssvc ...) by using 'allow dcerpc auth level connect:interface = yes' as option.

This option is over-ridden by the implementation specific restrictions. E.g. the drsuapi and backupkey protocols require DCERPC_AUTH_LEVEL_PRIVACY. The dnsserver protocol requires DCERPC_AUTH_LEVEL_INTEGRITY.

Default: allow dcerpc auth level connect = no

Example: allow dcerpc auth level connect = yes

allow dns updates (G)

This option determines what kind of updates to the DNS are allowed. DNS updates can either be disallowed completely by setting it to disabled, enabled over secure connections only by setting it to secure only or allowed in all cases by setting it to nonsecure.

Default: allow dns updates = secure only

Example: allow dns updates = disabled

allow insecure wide links (G)

In normal operation the option wide links which allows the server to follow symlinks outside of a share path is automatically disabled when unix extensions are enabled on a Samba server. This is done for security purposes to prevent UNIX clients creating symlinks to areas of the server file system that the administrator does not wish to export.

Setting allow insecure wide links to true disables the link between these two parameters, removing this protection and allowing a site to configure the server to follow symlinks (by setting wide links to "true") even when unix extensions is turned on.

It is not recommended to enable this option unless you fully understand the implications of allowing the server to follow symbolic links created by UNIX clients. For most normal Samba

configurations this would be considered a security hole and setting
this parameter is not recommended.

This option was added at the request of sites who had deliberately
set Samba up in this way and needed to continue supporting this
functionality without having to patch the Samba code.

Default: allow insecure wide links = no

allow nt4 crypto (G)

This option is deprecated and will be removed in future, as it is a
security problem if not set to "no" (which will be the hardcoded
behavior in future).

This option controls whether the netlogon server (currently only in
'active directory domain controller' mode), will reject clients
which do not support NETLOGON_NEG_STRONG_KEYS nor
NETLOGON_NEG_SUPPORTS_AES.

This option was added with Samba 4.2.0. It may lock out clients
which worked fine with Samba versions up to 4.1.x. as the effective
default was "yes" there, while it is "no" now.

If you have clients without RequireStrongKey = 1 in the registry,
you may need to set "allow nt4 crypto = yes", until you have fixed
all clients.

"allow nt4 crypto = yes" allows weak crypto to be negotiated, maybe
via downgrade attacks.

Avoid using this option! Use explicit 'allow nt4
crypto:COMPUTERACCOUNT = yes' instead! Which is available with the
patches for CVE-2022-38023 see
https://bugzilla.samba.org/show_bug.cgi?id=15240

Samba will log an error in the log files at log level 0 if legacy a
client is rejected or allowed without an explicit, 'allow nt4
crypto:COMPUTERACCOUNT = yes' option for the client. The message
will indicate the explicit 'allow nt4 crypto:COMPUTERACCOUNT = yes'
line to be added, if the legacy client software requires it. (The
log level can be adjusted with 'CVE_2022_38023:error_debug_level =
1' in order to complain only at a higher log level).

This allows admins to use "yes" only for a short grace period, in order to collect the explicit 'allow nt4 crypto:COMPUTERACCOUNT = yes' options.

This option is over-ridden by the effective value of 'yes' from the 'server reject md5 schannel:COMPUTERACCOUNT' and/or 'reject md5 clients' options.

Default: allow nt4 crypto = no

allow nt4 crypto:COMPUTERACCOUNT (G)

If you still have legacy domain members which required 'allow nt4 crypto = yes', it is possible to specify an explicit exception per computer account by using 'allow nt4 crypto:COMPUTERACCOUNT = yes' as option. Note that COMPUTERACCOUNT has to be the sAMAccountName value of the computer account (including the trailing '$' sign).

Samba will log a complaint in the log files at log level 0 about the security problem if the option is set to "yes", but the related computer does not require it. (The log level can be adjusted with 'CVE_2022_38023:warn_about_unused_debug_level = 1' in order to complain only at a higher log level).

Samba will log a warning in the log files at log level 5, if a setting is still needed for the specified computer account.

See CVE-2022-38023, https://bugzilla.samba.org/show_bug.cgi?id=15240.

This option overrides the allow nt4 crypto option.

This option is over-ridden by the effective value of 'yes' from the 'server reject md5 schannel:COMPUTERACCOUNT' and/or 'reject md5 clients' options.

Which means 'allow nt4 crypto:COMPUTERACCOUNT = yes' is only useful in combination with 'server reject md5 schannel:COMPUTERACCOUNT = no'

```
allow nt4 crypto:LEGACYCOMPUTER1$ = yes
server reject md5 schannel:LEGACYCOMPUTER1$ = no
allow nt4 crypto:NASBOX$ = yes
server reject md5 schannel:NASBOX$ = no
```

```
allow nt4 crypto:LEGACYCOMPUTER2$ = yes

server reject md5 schannel:LEGACYCOMPUTER2$ = no
```

No default

allow trusted domains (G)

This option only takes effect when the security option is set to server, domain or ads. If it is set to no, then attempts to connect to a resource from a domain or workgroup other than the one which smbd is running in will fail, even if that domain is trusted by the remote server doing the authentication.

This is useful if you only want your Samba server to serve resources to users in the domain it is a member of. As an example, suppose that there are two domains DOMA and DOMB. DOMB is trusted by DOMA, which contains the Samba server. Under normal circumstances, a user with an account in DOMB can then access the resources of a UNIX account with the same account name on the Samba server even if they do not have an account in DOMA. This can make implementing a security boundary difficult.

Default: allow trusted domains = yes

allow unsafe cluster upgrade (G)

If set to no (the default), smbd checks at startup if other smbd versions are running in the cluster and refuses to start if so.

This is done to protect data corruption in internal data structures due to incompatible Samba versions running concurrently in the same cluster. Setting this parameter to yes disables this safety check.

Default: allow unsafe cluster upgrade = no

apply group policies (G)

This option controls whether winbind will execute the gpupdate command defined in gpo update command on the Group Policy update interval. The Group Policy update interval is defined as every 90 minutes, plus a random offset between 0 and 30 minutes. This applies Group Policy Machine polices to the client or KDC and machine policies to a server.

Default: apply group policies = no

Example: apply group policies = yes

async dns timeout (G)

The number of seconds the asynchronous DNS resolver code in Samba will wait for responses. Some of the Samba client library code uses internal asynchronous DNS resolution for A and AAAA records when trying to find Active Directory Domain controllers. This value prevents this name resolution code from waiting for DNS server timeouts.

The minimum value of this parameter is clamped at 1 second.

Default: async dns timeout = 10

Example: async dns timeout = 20

async smb echo handler (G)

This parameter specifies whether Samba should fork the async smb echo handler. It can be beneficial if your file system can block syscalls for a very long time. In some circumstances, it prolongs the timeout that Windows uses to determine whether a connection is dead. This parameter is only for SMB1. For SMB2 and above TCP keepalives can be used instead.

Default: async smb echo handler = no

auth event notification (G)

When enabled, this option causes Samba (acting as an Active Directory Domain Controller) to stream authentication events across the internal message bus. Scripts built using Samba's python bindings can listen to these events by registering as the service auth_event.

This is not needed for the audit logging described in log level.

Instead, this should instead be considered a developer option (it assists in the Samba testsuite) rather than a facility for external auditing, as message delivery is not guaranteed (a feature that the testsuite works around).

The authentication events are also logged via the normal logging methods when the log level is set appropriately, say to auth_json_audit:3.

Default: auth event notification = no

preload

This parameter is a synonym for auto services.

auto services (G)

This is a list of services that you want to be automatically added to the browse lists. This is most useful for homes and printers services that would otherwise not be visible.

Note that if you just want all printers in your printcap file loaded then the load printers option is easier.

Default: auto services =

Example: auto services = fred lp colorlp

available (S)

This parameter lets you "turn off" a service. If available = no, then ALL attempts to connect to the service will fail. Such failures are logged.

Default: available = yes

bind dns directory

This parameter is a synonym for binddns dir.

binddns dir (G)

This parameters defines the directory samba will use to store the configuration files for bind, such as named.conf. NOTE: The bind dns directory needs to be on the same mount point as the private directory!

Default: binddns dir = /var/lib/samba/bind-dns

bind interfaces only (G)

This global parameter allows the Samba admin to limit what interfaces on a machine will serve SMB requests. It affects file service smbd(8) and name service nmbd(8) in a slightly different ways.

For name service it causes nmbd to bind to ports 137 and 138 on the interfaces listed in the interfaces parameter. nmbd also binds to the "all addresses" interface (0.0.0.0) on ports 137 and 138 for the purposes of reading broadcast messages. If this option is not set then nmbd will service name requests on all of these sockets.

If bind interfaces only is set then nmbd will check the source address of any packets coming in on the broadcast sockets and discard any that don't match the broadcast addresses of the interfaces in the interfaces parameter list. As unicast packets are received on the other sockets it allows nmbd to refuse to serve names to machines that send packets that arrive through any interfaces not listed in the interfaces list. IP Source address spoofing does defeat this simple check, however, so it must not be used seriously as a security feature for nmbd.

For file service it causes smbd(8) to bind only to the interface list given in the interfaces parameter. This restricts the networks that smbd will serve, to packets coming in on those interfaces.

Note that you should not use this parameter for machines that are serving PPP or other intermittent or non-broadcast network interfaces as it will not cope with non-permanent interfaces.

If bind interfaces only is set and the network address 127.0.0.1 is not added to the interfaces parameter list smbpasswd(8) may not work as expected due to the reasons covered below.

To change a users SMB password, the smbpasswd by default connects to the localhost - 127.0.0.1 address as an SMB client to issue the password change request. If bind interfaces only is set then unless the network address 127.0.0.1 is added to the interfaces parameter list then smbpasswd will fail to connect in it's default mode.

smbpasswd can be forced to use the primary IP interface of the local host by using its smbpasswd(8) -r remote machine parameter, with remote machine set to the IP name of the primary interface of the local host.

Default: bind interfaces only = no

blocking locks (S)

This parameter controls the behavior of smbd(8) when given a request by a client to obtain a byte range lock on a region of an open file, and the request has a time limit associated with it.

If this parameter is set and the lock range requested cannot be immediately satisfied, samba will internally queue the lock request, and periodically attempt to obtain the lock until the timeout period expires.

If this parameter is set to no, then samba will behave as previous versions of Samba would and will fail the lock request immediately if the lock range cannot be obtained.

Default: blocking locks = yes

block size (S)

This parameter controls the behavior of smbd(8) when reporting disk free sizes. By default, this reports a disk block size of 1024 bytes.

Changing this parameter may have some effect on the efficiency of client writes, this is not yet confirmed. This parameter was added to allow advanced administrators to change it (usually to a higher value) and test the effect it has on client write performance without re-compiling the code. As this is an experimental option it may be removed in a future release.

Changing this option does not change the disk free reporting size, just the block size unit reported to the client.

Default: block size = 1024

Example: block size = 4096

browsable

This parameter is a synonym for browseable.

browseable (S)

This controls whether this share is seen in the list of available shares in a net view and in the browse list.

Default: browseable = yes

browse list (G)

This controls whether smbd(8) will serve a browse list to a client doing a NetServerEnum call. Normally set to yes. You should never need to change this.

Default: browse list = yes

cache directory (G)

Usually, most of the TDB files are stored in the lock directory. Since Samba 3.4.0, it is possible to differentiate between TDB files with persistent data and TDB files with non-persistent data using the state directory and the cache directory options.

This option specifies the directory for storing TDB files containing non-persistent data that will be kept across service restarts. The directory should be placed on persistent storage, but the data can be safely deleted by an administrator.

Default: cache directory = /var/lib/samba

Example: cache directory = /var/run/samba/locks/cache

casesignames

This parameter is a synonym for case sensitive.

case sensitive (S)

See the discussion in the section name mangling.

Default: case sensitive = auto

change notify (G)

This parameter specifies whether Samba should reply to a client's file change notify requests.

You should never need to change this parameter

Default: change notify = yes

change share command (G)

Samba 2.2.0 introduced the ability to dynamically add and delete shares via the Windows NT 4.0 Server Manager. The change share command is used to define an external program or script which will modify an existing service definition in smb.conf.

In order to successfully execute the change share command, smbd requires that the administrator connects using a root account (i.e. uid == 0) or has the SeDiskOperatorPrivilege. Scripts defined in the change share command parameter are executed as root.

When executed, smbd will automatically invoke the change share command with six parameters.

? configFile - the location of the global smb.conf file.

? shareName - the name of the new share.

? pathName - path to an **existing** directory on disk.

? comment - comment string to associate with the new share.

? max connections Number of maximum simultaneous connections to this share.

? CSC policy - client side caching policy in string form. Valid values are: manual, documents, programs, disable.

This parameter is only used to modify existing file share definitions. To modify printer shares, use the "Printers..." folder as seen when browsing the Samba host.

Default: change share command =

Example: change share command = /usr/local/bin/changeshare

check parent directory delete on close (S)

A Windows SMB server prevents the client from creating files in a directory that has the delete-on-close flag set. By default Samba doesn't perform this check as this check is a quite expensive operation in Samba.

Default: check parent directory delete on close = no

check password script (G)

The name of a program that can be used to check password complexity. The password is sent to the program's standard input. The program must return 0 on a good password, or any other value if the password is bad. In case the password is considered weak (the program does not return 0) the user will be notified and the password change will fail.

In Samba AD, this script will be run AS ROOT by samba(8) without any substitutions.

Note that starting with Samba 4.11 the following environment variables are exported to the script:

? SAMBA_CPS_ACCOUNT_NAME is always present and contains the sAMAccountName of user, the is the same as the %u substitutions in the none AD DC case.

? SAMBA_CPS_USER_PRINCIPAL_NAME is optional in the AD DC case if the userPrincipalName is present.

? SAMBA_CPS_FULL_NAME is optional if the displayName is present.

Note: In the example directory is a sample program called crackcheck that uses cracklib to check the password quality.

Default: check password script = # Disabled

Example: check password script = /usr/local/sbin/crackcheck

cldap port (G)

This option controls the port used by the CLDAP protocol.

Default: cldap port = 389

Example: cldap port = 3389

client ipc max protocol (G)

The value of the parameter (a string) is the highest protocol level that will be supported for IPC$ connections as DCERPC transport.

Normally this option should not be set as the automatic negotiation phase in the SMB protocol takes care of choosing the appropriate protocol.

The value default refers to the latest supported protocol, currently SMB3_11.

See client max protocol for a full list of available protocols. The values CORE, COREPLUS, LANMAN1, LANMAN2 are silently upgraded to NT1.

Default: client ipc max protocol = default

Example: client ipc max protocol = SMB2_10

client ipc min protocol (G)

This setting controls the minimum protocol version that the will be attempted to use for IPC$ connections as DCERPC transport.

Normally this option should not be set as the automatic negotiation phase in the SMB protocol takes care of choosing the appropriate protocol.

The value default refers to the higher value of NT1 and the effective value of client min protocol.

See client max protocol for a full list of available protocols. The values CORE, COREPLUS, LANMAN1, LANMAN2 are silently upgraded to NT1.

Default: client ipc min protocol = default

Example: client ipc min protocol = SMB3_11

client ipc signing (G)

This controls whether the client is allowed or required to use SMB signing for IPC$ connections as DCERPC transport. Possible values are desired, required and disabled.

When set to required or default, SMB signing is mandatory.

When set to desired, SMB signing is offered, but not enforced and if set to disabled, SMB signing is not offered either.

Connections from winbindd to Active Directory Domain Controllers always enforce signing.

Default: client ipc signing = default

client lanman auth (G)

This parameter has been deprecated since Samba 4.13 and support for LanMan (as distinct from NTLM, NTLMv2 or Kerberos) authentication as a client will be removed in a future Samba release.

That is, in the future, the current default of client NTLMv2 auth = yes will be the enforced behaviour.

This parameter determines whether or not smbclient(8) and other samba client tools will attempt to authenticate itself to servers using the weaker LANMAN password hash. If disabled, only server which support NT password hashes (e.g. Windows NT/2000, Samba, etc... but not Windows 95/98) will be able to be connected from the Samba client.

The LANMAN encrypted response is easily broken, due to its case-insensitive nature, and the choice of algorithm. Clients without Windows 95/98 servers are advised to disable this option.

Disabling this option will also disable the client plaintext auth option.

Likewise, if the client ntlmv2 auth parameter is enabled, then only

NTLMv2 logins will be attempted.

Default: client lanman auth = no

client ldap sasl wrapping (G)

The client ldap sasl wrapping defines whether ldap traffic will be signed or signed and encrypted (sealed). Possible values are plain, sign and seal.

The values sign and seal are only available if Samba has been compiled against a modern OpenLDAP version (2.3.x or higher).

This option is needed in the case of Domain Controllers enforcing the usage of signed LDAP connections (e.g. Windows 2000 SP3 or higher). LDAP sign and seal can be controlled with the registry key "HKLM\System\CurrentControlSet\Services\NTDS\Parameters\LDAPServerIntegrity" on the Windows server side.

Depending on the used KRB5 library (MIT and older Heimdal versions) it is possible that the message "integrity only" is not supported.

In this case, sign is just an alias for seal.

The default value is sign. That implies synchronizing the time with the KDC in the case of using Kerberos.

Default: client ldap sasl wrapping = sign

client max protocol (G)

The value of the parameter (a string) is the highest protocol level that will be supported by the client.

Possible values are :

? CORE: Earliest version. No concept of user names.

? COREPLUS: Slight improvements on CORE for efficiency.

? LANMAN1: First modern version of the protocol. Long filename support.

? LANMAN2: Updates to Lanman1 protocol.

? NT1: Current up to date version of the protocol. Used by Windows NT. Known as CIFS.

? SMB2: Re-implementation of the SMB protocol. Used by Windows Vista and later versions of Windows. SMB2 has sub protocols available.

? SMB2_02: The earliest SMB2 version.

? SMB2_10: Windows 7 SMB2 version.

By default SMB2 selects the SMB2_10 variant.

? SMB3: The same as SMB2. Used by Windows 8. SMB3 has sub protocols available.

? SMB3_00: Windows 8 SMB3 version.

? SMB3_02: Windows 8.1 SMB3 version.

? SMB3_11: Windows 10 SMB3 version.

By default SMB3 selects the SMB3_11 variant.

Normally this option should not be set as the automatic negotiation phase in the SMB protocol takes care of choosing the appropriate protocol.

The value default refers to SMB3_11.

IPC$ connections for DCERPC e.g. in winbindd, are handled by the client ipc max protocol option.

Default: client max protocol = default

Example: client max protocol = LANMAN1

client min protocol (G)

This setting controls the minimum protocol version that the client will attempt to use.

Normally this option should not be set as the automatic negotiation phase in the SMB protocol takes care of choosing the appropriate protocol unless you connect to a legacy SMB1-only server.

See Related command: client max protocol for a full list of available protocols.

IPC$ connections for DCERPC e.g. in winbindd, are handled by the client ipc min protocol option.

Note that most command line tools support --option='client min protocol=NT1', so it may not be required to enable SMB1 protocols globally in smb.conf.

Default: client min protocol = SMB2_02

Example: client min protocol = NT1

client NTLMv2 auth (G)

This parameter has been deprecated since Samba 4.13 and support for NTLM and LanMan (as distinct from NTLMv2 or Kerberos authentication) will be removed in a future Samba release.

That is, in the future, the current default of client NTLMv2 auth = yes will be the enforced behaviour.

This parameter determines whether or not smbclient(8) will attempt to authenticate itself to servers using the NTLMv2 encrypted password response.

If enabled, only an NTLMv2 and LMv2 response (both much more secure than earlier versions) will be sent. Older servers (including NT4 < SP4, Win9x and Samba 2.2) are not compatible with NTLMv2 when not in an NTLMv2 supporting domain

Similarly, if enabled, NTLMv1, client lanman auth and client plaintext auth authentication will be disabled. This also disables share-level authentication.

If disabled, an NTLM response (and possibly a LANMAN response) will be sent by the client, depending on the value of client lanman auth.

Note that Windows Vista and later versions already use NTLMv2 by default, and some sites (particularly those following 'best practice' security polices) only allow NTLMv2 responses, and not the weaker LM or NTLM.

When client use spnego is also set to yes extended security (SPNEGO) is required in order to use NTLMv2 only within NTLMSSP. This behavior was introduced with the patches for CVE-2016-2111.

Default: client NTLMv2 auth = yes

client plaintext auth (G)

This parameter has been deprecated since Samba 4.13 and support for plaintext (as distinct from NTLM, NTLMv2 or Kerberos authentication) will be removed in a future Samba release.

That is, in the future, the current default of client plaintext auth = no will be the enforced behaviour.

Specifies whether a client should send a plaintext password if the server does not support encrypted passwords.

Default: client plaintext auth = no

client protection (G)

This parameter defines which protection Samba client tools should use by default.

Possible client settings are:

- ? default - Use the individual default values of the options:
  - ? client signing
  - ? client smb encrypt
- ? plain - This will send everything just as plaintext, signing or encryption are turned off.
- ? sign - This will enable integrity checking.
- ? encrypt - This will enable integrity checks and force encryption for privacy.

Default: client protection = default

client schannel (G)

This option is deprecated with Samba 4.8 and will be removed in future. At the same time the default changed to yes, which will be the hardcoded behavior in future.

This controls whether the client offers or even demands the use of the netlogon schannel. client schannel = no does not offer the schannel, client schannel = auto offers the schannel but does not enforce it, and client schannel = yes denies access if the server is not able to speak netlogon schannel.

Note that for active directory domains this is hardcoded to client schannel = yes.

This option is over-ridden by the require strong key option.

Default: client schannel = yes

Example: client schannel = auto

client signing (G)

This controls whether the client is allowed or required to use SMB signing. Possible values are desired, required and disabled.

When set to desired or default, SMB signing is offered, but not enforced.

When set to required, SMB signing is mandatory and if set to disabled, SMB signing is not offered either.

IPC$ connections for DCERPC e.g. in winbindd, are handled by the client ipc signing option.

Default: client signing = default

client smb encrypt (G)

This parameter controls whether a client should try or is required to use SMB encryption. It has different effects depending on whether the connection uses SMB1 or SMB3:

? If the connection uses SMB1, then this option controls the use of a Samba-specific extension to the SMB protocol introduced in Samba 3.2 that makes use of the Unix extensions.

? If the connection uses SMB2 or newer, then this option controls the use of the SMB-level encryption that is supported in SMB version 3.0 and above and available in Windows 8 and newer.

This parameter can be set globally. Possible values are off, if_required, desired, and required. A special value is default which is the implicit default setting of if_required.

Effects for SMB1

The Samba-specific encryption of SMB1 connections is an extension to the SMB protocol negotiated as part of the UNIX extensions. SMB encryption uses the GSSAPI (SSPI on Windows) ability to encrypt and sign every request/response in a SMB protocol stream. When enabled it provides a secure method of SMB/CIFS communication, similar to an ssh protected session, but using SMB/CIFS authentication to negotiate encryption and signing keys. Currently this is only supported smbclient of by Samba 3.2 and newer. Windows does not support this feature.

When set to default, SMB encryption is probed, but not enforced. When set to required, SMB encryption is required and if set to disabled, SMB encryption can not be negotiated.

Effects for SMB3 and newer

Native SMB transport encryption is available in SMB version 3.0 or newer. It is only used by Samba if client max protocol is set to SMB3 or newer.

These features can be controlled with settings of client smb encrypt as follows:

? Leaving it as default, explicitly setting default, or setting it to if_required globally will enable negotiation of encryption but will not turn on data encryption globally.

? Setting it to desired globally will enable negotiation and will turn on data encryption on sessions and share connections for those servers that support it.

? Setting it to required globally will enable negotiation and turn on data encryption on sessions and share connections. Clients that do not support encryption will be denied access to the server.

? Setting it to off globally will completely disable the encryption feature for all connections.

Default: client smb encrypt = default

client smb3 encryption algorithms (G)

This parameter specifies the availability and order of encryption algorithms which are available for negotiation in the SMB3_11 dialect.

It is also possible to remove individual algorithms from the default list, by prefixing them with '-'. This can avoid having to specify a hardcoded list.

Note: that the removal of AES-128-CCM from the list will result in SMB3_00 and SMB3_02 being unavailable, as it is the default and only available algorithm for these dialects.

Default: client smb3 encryption algorithms = AES-128-GCM,

AES-128-CCM, AES-256-GCM, AES-256-CCM

Example: client smb3 encryption algorithms = AES-256-GCM

Example: client smb3 encryption algorithms = -AES-128-GCM

-AES-128-CCM

client smb3 signing algorithms (G)

This parameter specifies the availability and order of signing algorithms which are available for negotiation in the SMB3_11 dialect.

It is also possible to remove individual algorithms from the default list, by prefixing them with '-'. This can avoid having to specify a hardcoded list.

Note: that the removal of AES-128-CMAC from the list will result in SMB3_00 and SMB3_02 being unavailable, and the removal of HMAC-SHA256 will result in SMB2_02 and SMB2_10 being unavailable, as these are the default and only available algorithms for these dialects.

Default: client smb3 signing algorithms = AES-128-GMAC,

AES-128-CMAC, HMAC-SHA256

Example: client smb3 signing algorithms = AES-128-CMAC, HMAC-SHA256

Example: client smb3 signing algorithms = -AES-128-CMAC

client use kerberos (G)

This parameter determines whether Samba client tools will try to authenticate using Kerberos. For Kerberos authentication you need to use dns names instead of IP addresses when connecting to a service.

Possible option settings are:

? desired - Kerberos authentication will be tried first and if it fails it automatically fallback to NTLM.

? required - Kerberos authentication will be required. There will be no falllback to NTLM or a different alternative.

? off - Don't use Kerberos, use NTLM instead or another alternative.

In case that weak cryptography is not allowed (e.g. FIPS mode) the default will be forced to required.

Default: client use kerberos = desired

client use spnego principal (G)

This parameter determines whether or not smbclient(8) and other samba components acting as a client will attempt to use the server-supplied principal sometimes given in the SPNEGO exchange.

If enabled, Samba can attempt to use Kerberos to contact servers known only by IP address. Kerberos relies on names, so ordinarily cannot function in this situation.

This is a VERY BAD IDEA for security reasons, and so this parameter SHOULD NOT BE USED. It will be removed in a future version of Samba.

If disabled, Samba will use the name used to look up the server when asking the KDC for a ticket. This avoids situations where a server may impersonate another, soliciting authentication as one principal while being known on the network as another.

Note that Windows XP SP2 and later versions already follow this behaviour, and Windows Vista and later servers no longer supply this 'rfc4178 hint' principal on the server side.

This parameter is deprecated in Samba 4.2.1 and will be removed (along with the functionality) in a later release of Samba.

Default: client use spnego principal = no

client use spnego (G)

This parameter has been deprecated since Samba 4.13 and support for NTLMv2, NTLM and LanMan authentication outside NTLMSSP will be removed in a future Samba release.

That is, in the future, the current default of client use spnego = yes will be the enforced behaviour.

This variable controls whether Samba clients will try to use Simple and Protected NEGOciation (as specified by rfc2478) with supporting servers (including WindowsXP, Windows2000 and Samba 3.0) to agree upon an authentication mechanism. This enables Kerberos authentication in particular.

When client NTLMv2 auth is also set to yes extended security (SPNEGO) is required in order to use NTLMv2 only within NTLMSSP. This behavior was introduced with the patches for CVE-2016-2111.

Default: client use spnego = yes

cluster addresses (G)

With this parameter you can add additional addresses that nmbd will register with a WINS server. Similarly, these addresses will be registered by default when net ads dns register is called with clustering = yes configured.

Default: cluster addresses =

Example: cluster addresses = 10.0.0.1 10.0.0.2 10.0.0.3

clustering (G)

This parameter specifies whether Samba should contact ctdb for accessing its tdb files and use ctdb as a backend for its messaging backend.

Set this parameter to yes only if you have a cluster setup with ctdb running.

Default: clustering = no

comment (S)

This is a text field that is seen next to a share when a client does a queries the server, either via the network neighborhood or via net view to list what shares are available.

If you want to set the string that is displayed next to the machine name then see the server string parameter.

Default: comment = # No comment

Example: comment = Fred's Files

config backend (G)

This controls the backend for storing the configuration. Possible values are file (the default) and registry. When config backend = registry is encountered while loading smb.conf, the configuration read so far is dropped and the global options are read from registry instead. So this triggers a registry only configuration.

Share definitions are not read immediately but instead registry shares is set to yes.

Note: This option can not be set inside the registry configuration itself.

Default: config backend = file

Example: config backend = registry

config file (G)

This allows you to override the config file to use, instead of the default (usually smb.conf). There is a chicken and egg problem here as this option is set in the config file!

For this reason, if the name of the config file has changed when the parameters are loaded then it will reload them from the new config file.

This option takes the usual substitutions, which can be very useful.

If the config file doesn't exist then it won't be loaded (allowing you to special case the config files of just a few clients).

No default

Example: config file = /usr/local/samba/lib/smb.conf.%m

copy (S)

This parameter allows you to "clone" service entries. The specified service is simply duplicated under the current service's name. Any parameters specified in the current section will override those in the section being copied.

This feature lets you set up a 'template' service and create similar services easily. Note that the service being copied must occur earlier in the configuration file than the service doing the copying.

Default: copy =

Example: copy = otherservice

create krb5 conf (G)

Setting this parameter to no prevents winbind from creating custom krb5.conf files. Winbind normally does this because the krb5 libraries are not AD-site-aware and thus would pick any domain controller out of potentially very many. Winbind is site-aware and makes the krb5 libraries use a local DC by creating its own krb5.conf files.

Preventing winbind from doing this might become necessary if you have to add special options into your system-krb5.conf that winbind does not see.

Default: create krb5 conf = yes

create mode

This parameter is a synonym for create mask.

create mask (S)

When a file is created, the necessary permissions are calculated according to the mapping from DOS modes to UNIX permissions, and the resulting UNIX mode is then bit-wise 'AND'ed with this parameter. This parameter may be thought of as a bit-wise MASK for the UNIX modes of a file. Any bit not set here will be removed from the modes set on a file when it is created.

The default value of this parameter removes the group and other write and execute bits from the UNIX modes.

Following this Samba will bit-wise 'OR' the UNIX mode created from this parameter with the value of the force create mode parameter which is set to 000 by default.

This parameter does not affect directory masks. See the parameter directory mask for details.

Default: create mask = 0744

Example: create mask = 0775

csc policy (S)

This stands for client-side caching policy, and specifies how clients capable of offline caching will cache the files in the share. The valid values are: manual, documents, programs, disable.

These values correspond to those used on Windows servers.

For example, shares containing roaming profiles can have offline caching disabled using csc policy = disable.

Default: csc policy = manual

Example: csc policy = programs

ctdbd socket (G)

If you set clustering=yes, you need to tell Samba where ctdbd listens on its unix domain socket. The default path as of ctdb 1.0 is /tmp/ctdb.socket which you have to explicitly set for Samba in smb.conf.

Default: ctdbd socket =

Example: ctdbd socket = /tmp/ctdb.socket

ctdb locktime warn threshold (G)

In a cluster environment using Samba and ctdb it is critical that locks on central ctdb-hosted databases like locking.tdb are not held for long. With the current Samba architecture it happens that Samba takes a lock and while holding that lock makes file system calls into the shared cluster file system. This option makes Samba warn if it detects that it has held locks for the specified number of milliseconds. If this happens, smbd will emit a debug level 0 message into its logs and potentially into syslog. The most likely reason for such a log message is that an operation of the cluster file system Samba exports is taking longer than expected. The messages are meant as a debugging aid for potential cluster problems.

The default value of 0 disables this logging.

Default: ctdb locktime warn threshold = 0

ctdb timeout (G)

This parameter specifies a timeout in milliseconds for the connection between Samba and ctdb. It is only valid if you have compiled Samba with clustering and if you have set clustering=yes.

When something in the cluster blocks, it can happen that we wait indefinitely long for ctdb, just adding to the blocking condition.

In a well-running cluster this should never happen, but there are too many components in a cluster that might have hickups. Choosing the right balance for this value is very tricky, because on a busy cluster long service times to transfer something across the cluster might be valid. Setting it too short will degrade the service your cluster presents, setting it too long might make the cluster itself not recover from something severely broken for too long.

Be aware that if you set this parameter, this needs to be in the file smb.conf, it is not really helpful to put this into a registry configuration (typical on a cluster), because to access the registry contact to ctdb is required.

Setting ctdb timeout to n makes any process waiting longer than n milliseconds for a reply by the cluster panic. Setting it to 0 (the default) makes Samba block forever, which is the highly recommended default.

Default: ctdb timeout = 0

cups connection timeout (G)

This parameter is only applicable if printing is set to cups.

If set, this option specifies the number of seconds that smbd will wait whilst trying to contact to the CUPS server. The connection will fail if it takes longer than this number of seconds.

Default: cups connection timeout = 30

Example: cups connection timeout = 60

cups encrypt (G)

This parameter is only applicable if printing is set to cups and if you use CUPS newer than 1.0.x.It is used to define whether or not Samba should use encryption when talking to the CUPS server.

Possible values are auto, yes and no

When set to auto we will try to do a TLS handshake on each CUPS connection setup. If that fails, we will fall back to unencrypted operation.

Default: cups encrypt = no

cups options (S)

This parameter is only applicable if printing is set to cups. Its value is a free form string of options passed directly to the cups library.

You can pass any generic print option known to CUPS (as listed in the CUPS "Software Users' Manual"). You can also pass any printer specific option (as listed in "lpoptions -d printername -l") valid for the target queue. Multiple parameters should be space-delimited name/value pairs according to the PAPI text option ABNF specification. Collection values ("name={a=... b=... c=...}") are stored with the curley brackets intact.

You should set this parameter to raw if your CUPS server error_log file contains messages such as "Unsupported format 'application/octet-stream'" when printing from a Windows client through Samba. It is no longer necessary to enable system wide raw printing in /etc/cups/mime.{convs,types}.

Default: cups options = ""

Example: cups options = "raw media=a4"

cups server (G)

This parameter is only applicable if printing is set to cups.

If set, this option overrides the ServerName option in the CUPS client.conf. This is necessary if you have virtual samba servers that connect to different CUPS daemons.

Optionally, a port can be specified by separating the server name and port number with a colon. If no port was specified, the default port for IPP (631) will be used.

Default: cups server = ""

Example: cups server = mycupsserver

Example: cups server = mycupsserver:1631

dcerpc endpoint servers (G)

Specifies which DCE/RPC endpoint servers should be run.

Default: dcerpc endpoint servers = epmapper, wkssvc, rpcecho, samr, netlogon, lsarpc, drsuapi, dssetup, unixinfo, browser, eventlog6, backupkey, dnsserver

Example: dcerpc endpoint servers = rpcecho

deadtime (G)

The value of the parameter (a decimal integer) represents the number of minutes of inactivity before a connection is considered dead, and it is disconnected. The deadtime only takes effect if the number of open files is zero.

This is useful to stop a server's resources being exhausted by a large number of inactive connections.

Most clients have an auto-reconnect feature when a connection is broken so in most cases this parameter should be transparent to users.

Using this parameter with a timeout of a few minutes is recommended for most systems.

A deadtime of zero indicates that no auto-disconnection should be performed.

Default: deadtime = 10080

Example: deadtime = 15

debug class (G)

With this boolean parameter enabled, the debug class (DBGC_CLASS) will be displayed in the debug header.

For more information about currently available debug classes, see section about log level.

Default: debug class = no

debug encryption (G)

This option will make the smbd server and client code using libsmb (smbclient, smbget, smbspool, ...) dump the Session Id, the decrypted Session Key, the Signing Key, the Application Key, the Encryption Key and the Decryption Key every time an SMB3+ session is established. This information will be printed in logs at level 0.

Warning: access to these values enables the decryption of any encrypted traffic on the dumped sessions. This option should only be enabled for debugging purposes.

Default: debug encryption = no

debug hires timestamp (G)

Sometimes the timestamps in the log messages are needed with a resolution of higher that seconds, this boolean parameter adds microsecond resolution to the timestamp message header when turned on.

Note that the parameter debug timestamp or debug syslog format must be on for this to have an effect.

Default: debug hires timestamp = yes

debug pid (G)

When using only one log file for more then one forked smbd(8)-process there may be hard to follow which process outputs which message. This boolean parameter is adds the process-id to the timestamp message headers in the logfile when turned on.

Note that the parameter debug timestamp must be on for this to have an effect.

Default: debug pid = no

debug prefix timestamp (G)

With this option enabled, the timestamp message header is prefixed to the debug message without the filename and function information that is included with the debug timestamp parameter. This gives timestamps to the messages without adding an additional line.

Note that this parameter overrides the debug timestamp parameter.

Default: debug prefix timestamp = no

debug syslog format (G)

With this option enabled, debug messages are printed in a single-line format like that traditionally produced by syslog. The timestamp consists of an abbreviated month, space-padded date, and time including seconds. This is followed by the hostname and the program name, with the process-ID in square brackets.

If debug hires timestamp is also enabled then an RFC5424 timestamp is used instead.

Default: debug syslog format = no

winbind debug traceid (G)

With this boolean parameter enabled, the per request unique traceid will be displayed in the debug header for winbind processes.

Default: winbind debug traceid = no

debug uid (G)

Samba is sometimes run as root and sometime run as the connected user, this boolean parameter inserts the current euid, egid, uid and gid to the timestamp message headers in the log file if turned on.

Note that the parameter debug timestamp must be on for this to have an effect.

Default: debug uid = no

dedicated keytab file (G)

Specifies the absolute path to the kerberos keytab file when kerberos method is set to "dedicated keytab".

Default: dedicated keytab file =

Example: dedicated keytab file = /usr/local/etc/krb5.keytab

default case (S)

See the section on name mangling. Also note the short preserve case parameter.

Default: default case = lower

default devmode (S)

This parameter is only applicable to printable services. When smbd is serving Printer Drivers to Windows NT/2k/XP clients, each printer on the Samba server has a Device Mode which defines things such as paper size and orientation and duplex settings. The device mode can only correctly be generated by the printer driver itself (which can only be executed on a Win32 platform). Because smbd is unable to execute the driver code to generate the device mode, the default behavior is to set this field to NULL.

Most problems with serving printer drivers to Windows NT/2k/XP clients can be traced to a problem with the generated device mode. Certain drivers will do things such as crashing the client's Explorer.exe with a NULL devmode. However, other printer drivers can cause the client's spooler service (spoolsv.exe) to die if the devmode was not created by the driver itself (i.e. smbd generates a default devmode).

This parameter should be used with care and tested with the printer driver in question. It is better to leave the device mode to NULL and let the Windows client set the correct values. Because drivers do not do this all the time, setting default devmode = yes will instruct smbd to generate a default one.

For more information on Windows NT/2k printing and Device Modes, see the MSDN documentation.

Default: default devmode = yes

default

This parameter is a synonym for default service.

default service (G)

This parameter specifies the name of a service which will be connected to if the service actually requested cannot be found.

Note that the square brackets are NOT given in the parameter value (see example below).

There is no default value for this parameter. If this parameter is not given, attempting to connect to a nonexistent service results in an error.

Typically the default service would be a guest ok, read-only service.

Also note that the apparent service name will be changed to equal that of the requested service, this is very useful as it allows you to use macros like %S to make a wildcard service.

Note also that any "_" characters in the name of the service used in the default service will get mapped to a "/". This allows for interesting things.

Default: default service =

Example: default service = pub

defer sharing violations (G)

Windows allows specifying how a file will be shared with other processes when it is opened. Sharing violations occur when a file is opened by a different process using options that violate the share settings specified by other processes. This parameter causes smbd to act as a Windows server does, and defer returning a "sharing violation" error message for up to one second, allowing the client to close the file causing the violation in the meantime.

UNIX by default does not have this behaviour.

There should be no reason to turn off this parameter, as it is designed to enable Samba to more correctly emulate Windows.

Default: defer sharing violations = yes

delete group script (G)

This is the full pathname to a script that will be run AS ROOT by smbd(8) when a group is requested to be deleted. It will expand any %g to the group name passed. This script is only useful for installations using the Windows NT domain administration tools.

Default: delete group script =

deleteprinter command (G)

With the introduction of MS-RPC based printer support for Windows NT/2000 clients in Samba 2.2, it is now possible to delete a printer at run time by issuing the DeletePrinter() RPC call.

For a Samba host this means that the printer must be physically deleted from the underlying printing system. The deleteprinter command defines a script to be run which will perform the necessary operations for removing the printer from the print system and from smb.conf.

The deleteprinter command is automatically called with only one parameter: printer name.

Once the deleteprinter command has been executed, smbd will reparse the smb.conf to check that the associated printer no longer exists.

If the sharename is still valid, then smbd will return an ACCESS_DENIED error to the client.

Default: deleteprinter command =

Example: deleteprinter command = /usr/bin/removeprinter

delete readonly (S)

This parameter allows readonly files to be deleted. This is not normal DOS semantics, but is allowed by UNIX.

This option may be useful for running applications such as rcs, where UNIX file ownership prevents changing file permissions, and DOS semantics prevent deletion of a read only file.

Default: delete readonly = no

delete share command (G)

Samba 2.2.0 introduced the ability to dynamically add and delete shares via the Windows NT 4.0 Server Manager. The delete share command is used to define an external program or script which will remove an existing service definition from smb.conf.

In order to successfully execute the delete share command, smbd requires that the administrator connects using a root account (i.e. uid == 0) or has the SeDiskOperatorPrivilege. Scripts defined in the delete share command parameter are executed as root.

When executed, smbd will automatically invoke the delete share command with two parameters.

? configFile - the location of the global smb.conf file.

? shareName - the name of the existing service.

This parameter is only used to remove file shares. To delete printer shares, see the deleteprinter command.

Default: delete share command =

Example: delete share command = /usr/local/bin/delshare

delete user from group script (G)

Full path to the script that will be called when a user is removed from a group using the Windows NT domain administration tools. It will be run by smbd(8) AS ROOT. Any %g will be replaced with the group name and any %u will be replaced with the user name.

Default: delete user from group script =

Example: delete user from group script = /usr/sbin/deluser %u %g

delete user script (G)

This is the full pathname to a script that will be run by smbd(8) when managing users with remote RPC (NT) tools.

This script is called when a remote client removes a user from the server, normally using 'User Manager for Domains' or rpcclient.

This script should delete the given UNIX username.

Default: delete user script =

Example: delete user script = /usr/local/samba/bin/del_user %u

delete veto files (S)

This option is used when Samba is attempting to delete a directory that contains one or more vetoed files or directories or non-visible files or directories (such as dangling symlinks that point nowhere). (see the veto files, hide special files, hide unreadable, hide unwriteable files options). If this option is set to no (the default) then if a vetoed directory contains any non-vetoed files or directories then the directory delete will fail. This is usually what you want.

If this option is set to yes, then Samba will attempt to recursively delete any files and directories within the vetoed directory. This can be useful for integration with file serving systems such as NetAtalk which create meta-files within directories you might normally veto DOS/Windows users from seeing (e.g. .AppleDouble)

Setting delete veto files = yes allows these directories to be transparently deleted when the parent directory is deleted (so long as the user has permissions to do so).

Default: delete veto files = no

dfree cache time (S)

The dfree cache time should only be used on systems where a problem occurs with the internal disk space calculations. This has been known to happen with Ultrix, but may occur with other operating systems. The symptom that was seen was an error of "Abort Retry Ignore" at the end of each directory listing.

This is a new parameter introduced in Samba version 3.0.21. It specifies in seconds the time that smbd will cache the output of a disk free query. If set to zero (the default) no caching is done.

This allows a heavily loaded server to prevent rapid spawning of dfree command scripts increasing the load.

By default this parameter is zero, meaning no caching will be done.

No default

Example: dfree cache time = 60

dfree command (S)

The dfree command setting should only be used on systems where a problem occurs with the internal disk space calculations. This has been known to happen with Ultrix, but may occur with other operating systems. The symptom that was seen was an error of "Abort Retry Ignore" at the end of each directory listing.

This setting allows the replacement of the internal routines to calculate the total disk space and amount available with an external routine. The example below gives a possible script that might fulfill this function.

In Samba version 3.0.21 this parameter has been changed to be a per-share parameter, and in addition the parameter dfree cache time was added to allow the output of this script to be cached for systems under heavy load.

The external program will be passed a single parameter indicating a directory in the filesystem being queried. This will typically consist of the string ./. The script should return two integers in ASCII. The first should be the total disk space in blocks, and the second should be the number of available blocks. An optional third return value can give the block size in bytes. The default blocksize is 1024 bytes.

Note: Your script should NOT be setuid or setgid and should be owned by (and writeable only by) root!

Where the script dfree (which must be made executable) could be:

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

or perhaps (on Sys V based systems):

```
#!/bin/sh
```

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

Note that you may have to replace the command names with full path names on some systems. Also note the arguments passed into the script should be quoted inside the script in case they contain special characters such as spaces or newlines.

By default internal routines for determining the disk capacity and remaining space will be used.

No default

Example: dfree command = /usr/local/samba/bin/dfree

dgram port (G)

Specifies which ports the server should listen on for NetBIOS datagram traffic.

Default: dgram port = 138

directory mode

This parameter is a synonym for directory mask.

directory mask (S)

This parameter is the octal modes which are used when converting DOS modes to UNIX modes when creating UNIX directories.

When a directory is created, the necessary permissions are calculated according to the mapping from DOS modes to UNIX permissions, and the resulting UNIX mode is then bit-wise 'AND'ed with this parameter. This parameter may be thought of as a bit-wise MASK for the UNIX modes of a directory. Any bit not set here will be removed from the modes set on a directory when it is created.

The default value of this parameter removes the 'group' and 'other' write bits from the UNIX mode, allowing only the user who owns the directory to modify it.

Following this Samba will bit-wise 'OR' the UNIX mode created from this parameter with the value of the force directory mode parameter. This parameter is set to 000 by default (i.e. no extra mode bits are added).

Default: directory mask = 0755

Example: directory mask = 0775

directory name cache size (S)

This parameter specifies the size of the directory name cache for SMB1 connections. It is not used for SMB2. It will be needed to turn this off for *BSD systems.

Default: directory name cache size = 100

directory security mask (S)

This parameter has been removed for Samba 4.0.0.

No default

disable netbios (G)

Enabling this parameter will disable netbios support in Samba. Netbios is the only available form of browsing in all windows versions except for 2000 and XP.

Note

Clients that only support netbios won't be able to see your samba server when netbios support is disabled.

Default: disable netbios = no

disable spoolss (G)

Enabling this parameter will disable Samba's support for the SPOOLSS set of MS-RPC's and will yield identical behavior as Samba 2.0.x. Windows NT/2000 clients will downgrade to using Lanman style printing commands. Windows 9x/ME will be unaffected by the parameter. However, this will also disable the ability to upload printer drivers to a Samba server via the Windows NT Add Printer Wizard or by using the NT printer properties dialog window. It will also disable the capability of Windows NT/2000 clients to download print drivers from the Samba host upon demand. Be very careful about enabling this parameter.

Default: disable spoolss = no

dmapi support (S)

This parameter specifies whether Samba should use DMAPI to determine whether a file is offline or not. This would typically be used in conjunction with a hierarchical storage system that automatically migrates files to tape.

Note that Samba infers the status of a file by examining the events that a DMAPI application has registered interest in. This heuristic is satisfactory for a number of hierarchical storage systems, but there may be system for which it will fail. In this case, Samba may erroneously report files to be offline.

This parameter is only available if a supported DMAPI implementation was found at compilation time. It will only be used if DMAPI is found to enabled on the system at run time.

Default: dmapi support = no

dns forwarder (G)

This option specifies the list of DNS servers that DNS requests will be forwarded to if they can not be handled by Samba itself.

The DNS forwarder is only used if the internal DNS server in Samba is used. Port numbers can be appended by separating them from the address by using a colon (':'). When specifying a port, IPv6 addresses must be enclosed in square brackets ('[' and ']'). IPv6 forwarder addresses with no port specified, don't need the square brackets, and default to port 53.

Default: dns forwarder =

Example: dns forwarder = 192.168.0.1 192.168.0.2 ::1 [2001:db8::1] [2001:db8:1:2::1]:54

dns port (G)

Specifies which ports the server should listen on for DNS traffic.

It makes possible to use another DNS server as a front and forward to Samba.

Warning

Dynamic DNS updates may not be proxied by the front DNS server when forwarding to Samba. Dynamic DNS update proxying depends on the features of the other DNS server used as a front.

Default: dns port = 53

dns proxy (G)

Specifies that nmbd(8) when acting as a WINS server and finding that a NetBIOS name has not been registered, should treat the

NetBIOS name word-for-word as a DNS name and do a lookup with the DNS server for that name on behalf of the name-querying client.

Note that the maximum length for a NetBIOS name is 15 characters, so the DNS name (or DNS alias) can likewise only be 15 characters, maximum.

nmbd spawns a second copy of itself to do the DNS name lookup requests, as doing a name lookup is a blocking action.

Default: dns proxy = yes

dns update command (G)

This option sets the command that is called when there are DNS updates. It should update the local machines DNS names using TSIG-GSS.

Default: dns update command =

/builddir/build/BUILD/samba-4.17.5/source4/scripting/bin/samba_dnsupdate

Example: dns update command = /usr/local/sbin/dnsupdate

dns zone scavenging (G)

When enabled (the default is disabled) unused dynamic dns records are periodically removed.

Warning

This option should not be enabled for installations created with versions of samba before 4.9. Doing this will result in the loss of static DNS entries. This is due to a bug in previous versions of samba (BUG 12451) which marked dynamic DNS records as static and static records as dynamic.

Note

If one record for a DNS name is static (non-aging) then no other record for that DNS name will be scavenged.

Default: dns zone scavenging = no

dns zone transfer clients allow (G)

This option specifies the list of IPs authorized to ask for dns zone transfer from bind DLZ module.

The IP list is comma and space separated and specified in the same syntax as used in hosts allow, specifically including IP address, IP prefixes and IP address masks.

As this is a DNS server option, hostnames are naturally not permitted.

The default behaviour is to deny any request. A request will be authorized only if the emitting client is identified in this list, and not in dns zone transfer clients deny

Default: dns zone transfer clients allow =

Example: dns zone transfer clients allow = 192.168.0.1

dns zone transfer clients deny (G)

This option specifies the list of IPs denied to ask for dns zone transfer from bind DLZ module.

The IP list is comma and space separated and specified in the same syntax as used in hosts allow, specifically including IP address, IP prefixes and IP address masks.

As this is a DNS server option, hostnames are naturally not permitted.

If a client identified in this list sends a zone transfer request, it will always be denied, even if they are in dns zone transfer clients allow. This allows the definition of specific denied clients within an authorized subnet.

Default: dns zone transfer clients deny =

Example: dns zone transfer clients deny = 192.168.0.1

domain logons (G)

This parameter has been deprecated since Samba 4.13 and support for NT4-style domain logons(as distinct from the Samba AD DC) will be removed in a future Samba release.

That is, in the future, the current default of domain logons = no will be the enforced behaviour.

If set to yes, the Samba server will provide the netlogon service for Windows 9X network logons for the workgroup it is in. This will also cause the Samba server to act as a domain controller for NT4 style domain services. For more details on setting up this feature see the Domain Control chapter of the Samba HOWTO Collection.

Default: domain logons = no

domain master (G)

Tell smbd(8) to enable WAN-wide browse list collation. Setting this option causes nmbd to claim a special domain specific NetBIOS name that identifies it as a domain master browser for its given workgroup. Local master browsers in the same workgroup on broadcast-isolated subnets will give this nmbd their local browse lists, and then ask smbd(8) for a complete copy of the browse list for the whole wide area network. Browser clients will then contact their local master browser, and will receive the domain-wide browse list, instead of just the list for their broadcast-isolated subnet.

Note that Windows NT Primary Domain Controllers expect to be able to claim this workgroup specific special NetBIOS name that identifies them as domain master browsers for that workgroup by default (i.e. there is no way to prevent a Windows NT PDC from attempting to do this). This means that if this parameter is set and nmbd claims the special name for a workgroup before a Windows NT PDC is able to do so then cross subnet browsing will behave strangely and may fail.

If domain logons = yes, then the default behavior is to enable the domain master parameter. If domain logons is not enabled (the default setting), then neither will domain master be enabled by default.

When domain logons = Yes the default setting for this parameter is Yes, with the result that Samba will be a PDC. If domain master = No, Samba will function as a BDC. In general, this parameter should be set to 'No' only on a BDC.

Default: domain master = auto

dont descend (S)

There are certain directories on some systems (e.g., the /proc tree under Linux) that are either not of interest to clients or are infinitely deep (recursive). This parameter allows you to specify a comma-delimited list of directories that the server should always show as empty.

Note that Samba can be very fussy about the exact format of the "dont descend" entries. For example you may need ./proc instead of just /proc. Experimentation is the best policy :-)

Default: dont descend =

Example: dont descend = /proc,/dev

dos charset (G)

DOS SMB clients assume the server has the same charset as they do. This option specifies which charset Samba should talk to DOS clients.

The default depends on which charsets you have installed. Samba tries to use charset 850 but falls back to ASCII in case it is not available. Run testparm(1) to check the default on your system.

No default

dos filemode (S)

The default behavior in Samba is to provide UNIX-like behavior where only the owner of a file/directory is able to change the permissions on it. However, this behavior is often confusing to DOS/Windows users. Enabling this parameter allows a user who has write access to the file (by whatever means, including an ACL permission) to modify the permissions (including ACL) on it. Note that a user belonging to the group owning the file will not be allowed to change permissions if the group is only granted read access. Ownership of the file/directory may also be changed. Note that using the VFS modules acl_xattr or acl_tdb which store native Windows as meta-data will automatically turn this option on for any share for which they are loaded, as they require this option to emulate Windows ACLs correctly.

Default: dos filemode = no

dos filetime resolution (S)

Under the DOS and Windows FAT filesystem, the finest granularity on time resolution is two seconds. Setting this parameter for a share causes Samba to round the reported time down to the nearest two second boundary when a query call that requires one second resolution is made to smbd(8).

This option is mainly used as a compatibility option for Visual C++ when used against Samba shares. If oplocks are enabled on a share, Visual C++ uses two different time reading calls to check if a file has changed since it was last read. One of these calls uses a one-second granularity, the other uses a two second granularity. As the two second call rounds any odd second down, then if the file has a timestamp of an odd number of seconds then the two timestamps will not match and Visual C++ will keep reporting the file has changed. Setting this option causes the two timestamps to match, and Visual C++ is happy.

Default: dos filetime resolution = no

dos filetimes (S)

Under DOS and Windows, if a user can write to a file they can change the timestamp on it. Under POSIX semantics, only the owner of the file or root may change the timestamp. By default, Samba emulates the DOS semantics and allows one to change the timestamp on a file if the user smbd is acting on behalf has write permissions. Due to changes in Microsoft Office 2000 and beyond, the default for this parameter has been changed from "no" to "yes" in Samba 3.0.14 and above. Microsoft Excel will display dialog box warnings about the file being changed by another user if this parameter is not set to "yes" and files are being shared between users.

Default: dos filetimes = yes

dsdb event notification (G)

When enabled, this option causes Samba (acting as an Active Directory Domain Controller) to stream Samba database events across the internal message bus. Scripts built using Samba's python bindings can listen to these events by registering as the service dsdb_event.

This is not needed for the audit logging described in log level.

Instead, this should instead be considered a developer option (it assists in the Samba testsuite) rather than a facility for external auditing, as message delivery is not guaranteed (a feature that the testsuite works around).

The Samba database events are also logged via the normal logging methods when the log level is set appropriately, say to dsdb_json_audit:5.

Default: dsdb event notification = no

dsdb group change notification (G)

When enabled, this option causes Samba (acting as an Active Directory Domain Controller) to stream group membership change events across the internal message bus. Scripts built using Samba's python bindings can listen to these events by registering as the service dsdb_group_event.

This is not needed for the audit logging described in log level.

Instead, this should instead be considered a developer option (it assists in the Samba testsuite) rather than a facility for external auditing, as message delivery is not guaranteed (a feature that the testsuite works around).

The Samba database events are also logged via the normal logging methods when the log level is set appropriately, say to dsdb_group_json_audit:5.

Default: dsdb group change notification = no

dsdb password event notification (G)

When enabled, this option causes Samba (acting as an Active Directory Domain Controller) to stream password change and reset events across the internal message bus. Scripts built using Samba's python bindings can listen to these events by registering as the service password_event.

This is not needed for the audit logging described in log level.

Instead, this should instead be considered a developer option (it assists in the Samba testsuite) rather than a facility for external auditing, as message delivery is not guaranteed (a feature that the testsuite works around).

The Samba database events are also logged via the normal logging methods when the log level is set appropriately, say to dsdb_password_json_audit:5.

Default: dsdb password event notification = no

durable handles (S)

This boolean parameter controls whether Samba can grant SMB2 durable file handles on a share.

Note that durable handles are only enabled if kernel oplocks = no, kernel share modes = no, and posix locking = no, i.e. if the share is configured for CIFS/SMB2 only access, not supporting interoperability features with local UNIX processes or NFS operations.

Also note that, for the time being, durability is not granted for a handle that has the delete on close flag set.

Default: durable handles = yes

ea support (S)

This boolean parameter controls whether smbd(8) will allow clients to attempt to access extended attributes on a share. In order to enable this parameter on a setup with default VFS modules:

? Samba must have been built with extended attributes support.

? The underlying filesystem exposed by the share must support extended attributes (e.g. the getfattr(1) / setfattr(1) utilities must work).

? Access to extended user attributes must be allowed by the underlying filesystem (e.g. when mounted with a system-dependent option like user_xattr on Linux).

This option exposes the "user" attribute namespace from the underlying filesystem to clients. In order to match Windows conventions, the namespace prefix ("user.") is stripped from the attribute name on the client side. The handling of further attribute namespaces (like "security", "system", or "trusted") is not affected by this option.

Note that the SMB protocol allows setting attributes whose value is 64K bytes long, and that on NTFS, the maximum storage space for extended attributes per file is 64K. On most UNIX systems (Solaris and ZFS file system being the exception), the limits are much lower - typically 4K. Worse, the same 4K space is often used to store system metadata such as POSIX ACLs, or Samba's NT ACLs. Giving clients access to this tight space via extended attribute support could consume all of it by unsuspecting client applications, which would prevent changing system metadata due to lack of space. The default has changed to yes in Samba release 4.9.0 and above to allow better Windows fileserver compatibility in a default install.

Default: ea support = yes

elasticsearch:address (S)

Specifies the name of the Elasticsearch server to use for Spotlight queries when using the Elasticsearch backend.

Default: elasticsearch:address = localhost

Example: elasticsearch:address = needle.haystack.samba.org

elasticsearch:ignore unknown attribute (G)

Ignore unknown Spotlight attributes in search queries. An example query using the unsupported attribute "kMDItemTopic" would be kMDItemTopic=="hotstuff". By default any query using such a type would completely fail. By enabling this option, if the type match is a subexpression of a larger expression, then this subexpression is just ignored.

Default: elasticsearch:ignore unknown attribute = no

Example: elasticsearch:ignore unknown attribute = yes

elasticsearch:ignore unknown type (G)

Ignore unknown Spotlight types in search queries. An example query using the unsupported type "public.calendar-event" would be kMDItemContentType=="public.calendar-event". By default any query using such a type would completely fail. By enabling this option, if the type match is a subexpression of a larger expression, then this subexpression is just ignored.

Default: elasticsearch:ignore unknown type = no

Example: elasticsearch:ignore unknown type = yes

elasticsearch:index (S)

Specifies the name of the Elasticsearch index to use for Spotlight queries when using the Elasticsearch backend. The default value of "_all" is a special Elasticsearch value that performs the search operation on all indices.

Default: elasticsearch:index = _all

Example: elasticsearch:index = spotlight

elasticsearch:mappings (G)

Path to a file specifying metadata attribute mappings in JSON format. Use by the Elasticsearch backend of the Spotlight RPC service.

Default: elasticsearch:mappings = /usr/share/samba/elasticsearch_mappings.json

Example: elasticsearch:mappings = /usr/share/foo/mymappings.json

elasticsearch:max results (S)

Path to a file specifying metadata attribute mappings in JSON format. Used by the Elasticsearch backend of the Spotlight RPC service. A value of 0 means no limit.

Default: elasticsearch:max results = 100

Example: elasticsearch:max results = 10

elasticsearch:port (S)

Specifies the TCP port of the Elasticsearch server to use for Spotlight queries when using the Elasticsearch backend.

Default: elasticsearch:port = 9200

Example: elasticsearch:port = 9201

elasticsearch:use tls (S)

Specifies whether to use HTTPS when talking to the Elasticsearch server used for Spotlight queries when using the Elasticsearch backend.

Default: elasticsearch:use tls = no

Example: elasticsearch:use tls = yes

enable asu support (G)

Hosts running the "Advanced Server for Unix (ASU)" product require some special accommodations such as creating a builtin [ADMIN$] share that only supports IPC connections. The has been the default behavior in smbd for many years. However, certain Microsoft applications such as the Print Migrator tool require that the remote server support an [ADMIN$] file share. Disabling this parameter allows for creating an [ADMIN$] file share in smb.conf.

Default: enable asu support = no

enable core files (G)

This parameter specifies whether core dumps should be written on internal exits. Normally set to yes. You should never need to change this.

Default: enable core files = yes

Example: enable core files = no

enable privileges (G)

This deprecated parameter controls whether or not smbd will honor privileges assigned to specific SIDs via either net rpc rights or one of the Windows user and group manager tools. This parameter is enabled by default. It can be disabled to prevent members of the Domain Admins group from being able to assign privileges to users or groups which can then result in certain smbd operations running as root that would normally run under the context of the connected user.

An example of how privileges can be used is to assign the right to join clients to a Samba controlled domain without providing root access to the server via smbd.

Please read the extended description provided in the Samba HOWTO documentation.

Default: enable privileges = yes

enable spoolss (G)

Inverted synonym for disable spoolss.

Default: enable spoolss = yes

encrypt passwords (G)

This parameter has been deprecated since Samba 4.11 and support for plaintext (as distinct from NTLM, NTLMv2 or Kerberos authentication) will be removed in a future Samba release.

That is, in the future, the current default of encrypt passwords = yes will be the enforced behaviour.

This boolean controls whether encrypted passwords will be negotiated with the client. Note that Windows NT 4.0 SP3 and above and also Windows 98 will by default expect encrypted passwords unless a registry entry is changed. To use encrypted passwords in Samba see the chapter "User Database" in the Samba HOWTO Collection.

MS Windows clients that expect Microsoft encrypted passwords and that do not have plain text password support enabled will be able to connect only to a Samba server that has encrypted password support enabled and for which the user accounts have a valid encrypted password. Refer to the smbpasswd command man page for information regarding the creation of encrypted passwords for user accounts.

The use of plain text passwords is NOT advised as support for this feature is no longer maintained in Microsoft Windows products. If you want to use plain text passwords you must set this parameter to no.

In order for encrypted passwords to work correctly smbd(8) must either have access to a local smbpasswd(5) file (see the smbpasswd(8) program for information on how to set up and maintain this file), or set the security = [domain|ads] parameter which causes smbd to authenticate against another server.

Default: encrypt passwords = yes

enhanced browsing (G)

This option enables a couple of enhancements to cross-subnet browse propagation that have been added in Samba but which are not standard in Microsoft implementations.

The first enhancement to browse propagation consists of a regular wildcard query to a Samba WINS server for all Domain Master Browsers, followed by a browse synchronization with each of the returned DMBs. The second enhancement consists of a regular randomised browse synchronization with all currently known DMBs.

You may wish to disable this option if you have a problem with empty workgroups not disappearing from browse lists. Due to the restrictions of the browse protocols, these enhancements can cause a empty workgroup to stay around forever which can be annoying.

In general you should leave this option enabled as it makes cross-subnet browse propagation much more reliable.

Default: enhanced browsing = yes

enumports command (G)

The concept of a "port" is fairly foreign to UNIX hosts. Under Windows NT/2000 print servers, a port is associated with a port monitor and generally takes the form of a local port (i.e. LPT1:, COM1:, FILE:) or a remote port (i.e. LPD Port Monitor, etc...). By default, Samba has only one port defined--"Samba Printer Port". Under Windows NT/2000, all printers must have a valid port name. If you wish to have a list of ports displayed (smbd does not use a port name for anything) other than the default "Samba Printer Port", you can define enumports command to point to a program which should generate a list of ports, one per line, to standard output. This listing will then be used in response to the level 1 and 2 EnumPorts() RPC.

Default: enumports command =

Example: enumports command = /usr/bin/listports

eventlog list (G)

This option defines a list of log names that Samba will report to the Microsoft EventViewer utility. The listed eventlogs will be associated with tdb file on disk in the $(statedir)/eventlog.

The administrator must use an external process to parse the normal

Unix logs such as /var/log/messages and write then entries to the eventlog tdb files. Refer to the eventlogadm(8) utility for how to write eventlog entries.

Default: eventlog list =

Example: eventlog list = Security Application Syslog Apache

fake directory create times (S)

NTFS and Windows VFAT file systems keep a create time for all files and directories. This is not the same as the ctime - status change time - that Unix keeps, so Samba by default reports the earliest of the various times Unix does keep. Setting this parameter for a share causes Samba to always report midnight 1-1-1980 as the create time for directories.

This option is mainly used as a compatibility option for Visual C++ when used against Samba shares. Visual C++ generated makefiles have the object directory as a dependency for each object file, and a make rule to create the directory. Also, when NMAKE compares timestamps it uses the creation time when examining a directory. Thus the object directory will be created if it does not exist, but once it does exist it will always have an earlier timestamp than the object files it contains.

However, Unix time semantics mean that the create time reported by Samba will be updated whenever a file is created or deleted in the directory. NMAKE finds all object files in the object directory. The timestamp of the last one built is then compared to the timestamp of the object directory. If the directory's timestamp if newer, then all object files will be rebuilt. Enabling this option ensures directories always predate their contents and an NMAKE build will proceed as expected.

Default: fake directory create times = no

fake oplocks (S)

Oplocks are the way that SMB clients get permission from a server to locally cache file operations. If a server grants an oplock (opportunistic lock) then the client is free to assume that it is the only one accessing the file and it will aggressively cache file data. With some oplock types the client may even cache file open/close operations. This can give enormous performance benefits.

When you set fake oplocks = yes, smbd(8) will always grant oplock requests no matter how many clients are using the file.

It is generally much better to use the real oplocks support rather than this parameter.

If you enable this option on all read-only shares or shares that you know will only be accessed from one client at a time such as physically read-only media like CDROMs, you will see a big performance improvement on many operations. If you enable this option on shares where multiple clients may be accessing the files read-write at the same time you can get data corruption. Use this option carefully!

Default: fake oplocks = no

follow symlinks (S)

This parameter allows the Samba administrator to stop smbd(8) from following symbolic links in a particular share. Setting this parameter to no prevents any file or directory that is a symbolic link from being followed (the user will get an error). This option is very useful to stop users from adding a symbolic link to /etc/passwd in their home directory for instance. However it will slow filename lookups down slightly.

This option is enabled (i.e. smbd will follow symbolic links) by default.

Default: follow symlinks = yes

smbd force process locks (S)

This boolean option tells smbd whether to forcefully disable the use of Open File Description locks on Linux.

This option should not be changed from the default unless you know what you're doing.

Default: smbd force process locks = no

force create mode (S)

This parameter specifies a set of UNIX mode bit permissions that will always be set on a file created by Samba. This is done by bitwise 'OR'ing these bits onto the mode bits of a file that is being created. The default for this parameter is (in octal) 000.

The modes in this parameter are bitwise 'OR'ed onto the file mode after the mask set in the create mask parameter is applied.

The example below would force all newly created files to have read and execute permissions set for 'group' and 'other' as well as the read/write/execute bits set for the 'user'.

Default: force create mode = 0000

Example: force create mode = 0755

force directory mode (S)

This parameter specifies a set of UNIX mode bit permissions that will always be set on a directory created by Samba. This is done by bitwise 'OR'ing these bits onto the mode bits of a directory that is being created. The default for this parameter is (in octal) 0000 which will not add any extra permission bits to a created directory. This operation is done after the mode mask in the parameter directory mask is applied.

The example below would force all created directories to have read and execute permissions set for 'group' and 'other' as well as the read/write/execute bits set for the 'user'.

Default: force directory mode = 0000

Example: force directory mode = 0755

force directory security mode (S)

This parameter has been removed for Samba 4.0.0.

No default

group

This parameter is a synonym for force group.

force group (S)

This specifies a UNIX group name that will be assigned as the default primary group for all users connecting to this service.

This is useful for sharing files by ensuring that all access to files on service will use the named group for their permissions checking. Thus, by assigning permissions for this group to the files and directories within this service the Samba administrator can restrict or allow sharing of these files.

In Samba 2.0.5 and above this parameter has extended functionality in the following way. If the group name listed here has a '+' character prepended to it then the current user accessing the share only has the primary group default assigned to this group if they are already assigned as a member of that group. This allows an administrator to decide that only users who are already in a particular group will create files with group ownership set to that group. This gives a finer granularity of ownership assignment. For example, the setting force group = +sys means that only users who are already in group sys will have their default primary group assigned to sys when accessing this Samba share. All other users will retain their ordinary primary group.

If the force user parameter is also set the group specified in force group will override the primary group set in force user.

Default: force group =

Example: force group = agroup

force printername (S)

When printing from Windows NT (or later), each printer in smb.conf has two associated names which can be used by the client. The first is the sharename (or shortname) defined in smb.conf. This is the only printername available for use by Windows 9x clients. The second name associated with a printer can be seen when browsing to the "Printers" (or "Printers and Faxes") folder on the Samba server. This is referred to simply as the printername (not to be confused with the printer name option).

When assigning a new driver to a printer on a remote Windows compatible print server such as Samba, the Windows client will rename the printer to match the driver name just uploaded. This can result in confusion for users when multiple printers are bound to the same driver. To prevent Samba from allowing the printer's printername to differ from the sharename defined in smb.conf, set force printername = yes.

Be aware that enabling this parameter may affect migrating printers from a Windows server to Samba since Windows has no way to force the sharename and printername to match.

It is recommended that this parameter's value not be changed once the printer is in use by clients as this could cause a user not be able to delete printer connections from their local Printers folder.

Default: force printername = no

force security mode (S)

This parameter has been removed for Samba 4.0.0.

No default

force unknown acl user (S)

If this parameter is set, a Windows NT ACL that contains an unknown SID (security descriptor, or representation of a user or group id) as the owner or group owner of the file will be silently mapped into the current UNIX uid or gid of the currently connected user.

This is designed to allow Windows NT clients to copy files and folders containing ACLs that were created locally on the client machine and contain users local to that machine only (no domain users) to be copied to a Samba server (usually with XCOPY /O) and have the unknown userid and groupid of the file owner map to the current connected user. This can only be fixed correctly when winbindd allows arbitrary mapping from any Windows NT SID to a UNIX uid or gid.

Try using this parameter when XCOPY /O gives an ACCESS_DENIED error.

Default: force unknown acl user = no

force user (S)

This specifies a UNIX user name that will be assigned as the default user for all users connecting to this service. This is useful for sharing files. You should also use it carefully as using it incorrectly can cause security problems.

This user name only gets used once a connection is established. Thus clients still need to connect as a valid user and supply a valid password. Once connected, all file operations will be performed as the "forced user", no matter what username the client connected as. This can be very useful.

In Samba 2.0.5 and above this parameter also causes the primary group of the forced user to be used as the primary group for all file activity. Prior to 2.0.5 the primary group was left as the primary group of the connecting user (this was a bug).

Default: force user =

Example: force user = auser

fss: prune stale (G)

When enabled, Samba's File Server Remote VSS Protocol (FSRVP) server checks all FSRVP initiated snapshots on startup, and removes any corresponding state (including share definitions) for nonexistent snapshot paths.

Default: fss: prune stale = no

Example: fss: prune stale = yes

fss: sequence timeout (G)

The File Server Remote VSS Protocol (FSRVP) server includes a message sequence timer to ensure cleanup on unexpected client disconnect. This parameter overrides the default timeout between FSRVP operations. FSRVP timeouts can be completely disabled via a value of 0.

Default: fss: sequence timeout = 180 or 1800, depending on operation

Example: fss: sequence timeout = 0

fstype (S)

This parameter allows the administrator to configure the string that specifies the type of filesystem a share is using that is reported by smbd(8) when a client queries the filesystem type for a share. The default type is NTFS for compatibility with Windows NT but this can be changed to other strings such as Samba or FAT if required.

Default: fstype = NTFS

Example: fstype = Samba

get quota command (G)

The get quota command should only be used whenever there is no operating system API available from the OS that samba can use.

This option is only available Samba was compiled with quotas support.

This parameter should specify the path to a script that queries the quota information for the specified user/group for the partition that the specified directory is on.

Such a script is being given 3 arguments:

? directory

? type of query

? uid of user or gid of group

The directory is actually mostly just "." - It needs to be treated relatively to the current working directory that the script can also query.

The type of query can be one of:

? 1 - user quotas

? 2 - user default quotas (uid = -1)

? 3 - group quotas

? 4 - group default quotas (gid = -1)

This script should print one line as output with spaces between the columns. The printed columns should be:

? 1 - quota flags (0 = no quotas, 1 = quotas enabled, 2 = quotas enabled and enforced)

? 2 - number of currently used blocks

? 3 - the softlimit number of blocks

? 4 - the hardlimit number of blocks

? 5 - currently used number of inodes

? 6 - the softlimit number of inodes

? 7 - the hardlimit number of inodes

? 8 (optional) - the number of bytes in a block(default is 1024)

Default: get quota command =

Example: get quota command = /usr/local/sbin/query_quota

getwd cache (G)

This is a tuning option. When this is enabled a caching algorithm will be used to reduce the time taken for getwd() calls. This can have a significant impact on performance, especially when the wide links parameter is set to no.

Default: getwd cache = yes

gpo update command (G)

This option sets the command that is called to apply GPO policies. The samba-gpupdate script applies System Access and Kerberos Policies to the KDC. System Access policies set minPwdAge, maxPwdAge, minPwdLength, and pwdProperties in the samdb. Kerberos Policies set kdc:service ticket lifetime, kdc:user ticket lifetime, and kdc:renewal lifetime in smb.conf.

Default: gpo update command =

/builddir/build/BUILD/samba-4.17.5/source4/scripting/bin/samba-gpupdate

Example: gpo update command = /usr/local/sbin/gpoupdate

guest account (G)

This is a username which will be used for access to services which are specified as guest ok (see below). Whatever privileges this user has will be available to any client connecting to the guest service. This user must exist in the password file, but does not require a valid login. The user account "ftp" is often a good choice for this parameter.

On some systems the default guest account "nobody" may not be able to print. Use another account in this case. You should test this by trying to log in as your guest user (perhaps by using the su - command) and trying to print using the system print command such as lpr(1) or lp(1).

This parameter does not accept % macros, because many parts of the system require this value to be constant for correct operation.

Default: guest account = nobody # default can be changed at compile-time

Example: guest account = ftp

public

This parameter is a synonym for guest ok.

guest ok (S)

If this parameter is yes for a service, then no password is required to connect to the service. Privileges will be those of the guest account.

This parameter nullifies the benefits of setting restrict anonymous = 2

See the section below on security for more information about this option.

Default: guest ok = no

only guest

This parameter is a synonym for guest only.

guest only (S)

If this parameter is yes for a service, then only guest connections to the service are permitted. This parameter will have no effect if guest ok is not set for the service.

See the section below on security for more information about this option.

Default: guest only = no

hide dot files (S)

This is a boolean parameter that controls whether files starting with a dot appear as hidden files.

Default: hide dot files = yes

hide files (S)

This is a list of files or directories that are not visible but are accessible. The DOS 'hidden' attribute is applied to any files or directories that match.

Each entry in the list must be separated by a '/', which allows spaces to be included in the entry. '*' and '?' can be used to specify multiple files or directories as in DOS wildcards.

Each entry must be a Unix path, not a DOS path and must not include the Unix directory separator '/'.

Note that the case sensitivity option is applicable in hiding files.

Setting this parameter will affect the performance of Samba, as it will be forced to check all files and directories for a match as they are scanned.

The example shown above is based on files that the Macintosh SMB client (DAVE) available from Thursby creates for internal use, and also still hides all files beginning with a dot.

An example of us of this parameter is:

```
hide files = /.*/DesktopFolderDB/TrashFor%m/resource.frk/
```

Default: hide files = # no file are hidden

hide new files timeout (S)

Setting this parameter to something but 0 hides files that have been modified less than N seconds ago.

It can be used for ingest/process queue style workloads. A processing application should only see files that are definitely finished. As many applications do not have proper external workflow control, this can be a way to make sure processing does not interfere with file ingest.

Default: hide new files timeout = 0

hide special files (S)

This parameter prevents clients from seeing special files such as sockets, devices and fifo's in directory listings.

Default: hide special files = no

hide unreadable (S)

This parameter prevents clients from seeing the existence of files that cannot be read. Defaults to off.

Please note that enabling this can slow down listing large directories significantly. Samba has to evaluate the ACLs of all directory members, which can be a lot of effort.

Default: hide unreadable = no

hide unwriteable files (S)

This parameter prevents clients from seeing the existence of files that cannot be written to. Defaults to off. Note that unwriteable directories are shown as usual.

Please note that enabling this can slow down listing large directories significantly. Samba has to evaluate the ACLs of all directory members, which can be a lot of effort.

Default: hide unwriteable files = no

honor change notify privilege (S)

This option can be used to make use of the change notify privilege. By default notify results are not checked against the file system permissions.

If "honor change notify privilege" is enabled, a user will only receive notify results, if he has change notify privilege or sufficient file system permissions. If a user has the change notify privilege, he will receive all requested notify results, even if the user does not have the permissions on the file system.

Default: honor change notify privilege = no

host msdfs (G)

If set to yes, Samba will act as a Dfs server, and allow Dfs-aware clients to browse Dfs trees hosted on the server.

See also the msdfs root share level parameter. For more information on setting up a Dfs tree on Samba, refer to the MSFDS chapter in the book Samba3-HOWTO.

Default: host msdfs = yes

hostname lookups (G)

Specifies whether samba should use (expensive) hostname lookups or use the ip addresses instead. An example place where hostname lookups are currently used is when checking the hosts deny and hosts allow.

Default: hostname lookups = no

Example: hostname lookups = yes

allow hosts

This parameter is a synonym for hosts allow.

hosts allow (S)

A synonym for this parameter is allow hosts.

This parameter is a comma, space, or tab delimited set of hosts which are permitted to access a service.

If specified in the [global] section then it will apply to all services, regardless of whether the individual service has a different setting.

You can specify the hosts by name or IP number. For example, you could restrict access to only the hosts on a Class C subnet with something like allow hosts = 150.203.5.. The full syntax of the list is described in the man page hosts_access(5). Note that this man page may not be present on your system, so a brief description will be given here also.

Note that the localhost address 127.0.0.1 will always be allowed access unless specifically denied by a hosts deny option.

You can also specify hosts by network/netmask pairs and by netgroup names if your system supports netgroups. The EXCEPT keyword can also be used to limit a wildcard list. The following examples may provide some help:

Example 1: allow all IPs in 150.203.*.*; except one

hosts allow = 150.203. EXCEPT 150.203.6.66

Example 2: allow hosts that match the given network/netmask

hosts allow = 150.203.15.0/255.255.255.0

Example 3: allow a couple of hosts

hosts allow = lapland, arvidsjaur

Example 4: allow only hosts in NIS netgroup "foonet", but deny access from one particular host

hosts allow = @foonet

hosts deny = pirate

Note

Note that access still requires suitable user-level passwords.

See testparm(1) for a way of testing your host access to see if it does what you expect.

Default: hosts allow = # none (i.e., all hosts permitted access)

Example: hosts allow = 150.203.5. myhost.mynet.edu.au

deny hosts

This parameter is a synonym for hosts deny.

hosts deny (S)

The opposite of hosts allow - hosts listed here are NOT permitted access to services unless the specific services have their own lists to override this one. Where the lists conflict, the allow list takes precedence.

In the event that it is necessary to deny all by default, use the keyword ALL (or the netmask 0.0.0.0/0) and then explicitly specify to the hosts allow = hosts allow parameter those hosts that should be permitted access.

Default: hosts deny = # none (i.e., no hosts specifically excluded)

Example: hosts deny = 150.203.4. badhost.mynet.edu.au

idmap backend (G)

The idmap backend provides a plugin interface for Winbind to use varying backends to store SID/uid/gid mapping tables.

This option specifies the default backend that is used when no special configuration set, but it is now deprecated in favour of the new spelling idmap config * : backend.

Default: idmap backend = tdb

idmap cache time (G)

This parameter specifies the number of seconds that Winbind's idmap interface will cache positive SID/uid/gid query results. By default, Samba will cache these results for one week.

Default: idmap cache time = 604800

idmap config DOMAIN : OPTION (G)

ID mapping in Samba is the mapping between Windows SIDs and Unix user and group IDs. This is performed by Winbindd with a configurable plugin interface. Samba's ID mapping is configured by options starting with the idmap config prefix. An idmap option consists of the idmap config prefix, followed by a domain name or the asterisk character (*), a colon, and the name of an idmap setting for the chosen domain.

The idmap configuration is hence divided into groups, one group for each domain to be configured, and one group with the asterisk instead of a proper domain name, which specifies the default configuration that is used to catch all domains that do not have an explicit idmap configuration of their own.

There are three general options available:

backend = backend_name

This specifies the name of the idmap plugin to use as the SID/uid/gid backend for this domain. The standard backends are tdb (idmap_tdb(8)), tdb2 (idmap_tdb2(8)), ldap (idmap_ldap(8)), rid (idmap_rid(8)), hash (idmap_hash(8)), autorid (idmap_autorid(8)), ad (idmap_ad(8)) and nss (idmap_nss(8)). The corresponding manual pages contain the details, but here is a summary.

The first three of these create mappings of their own using internal unixid counters and store the mappings in a database. These are suitable for use in the default idmap configuration. The rid and hash backends use a pure algorithmic calculation to determine the unixid for a SID. The autorid module is a mixture of the tdb and rid backend. It creates ranges for each domain encountered and then uses the rid algorithm for each of these automatically configured domains individually. The ad backend uses unix ids stored in Active Directory via the standard schema extensions. The nss backend reverses the standard winbindd setup and gets the unix ids via names from nsswitch which can be useful in an ldap setup.

range = low - high

Defines the available matching uid and gid range for which the backend is authoritative. For allocating backends, this also defines the start and the end of the range for allocating new unique IDs.

winbind uses this parameter to find the backend that is authoritative for a unix ID to SID mapping, so it must be set for each individually configured domain and for the default configuration. The configured ranges must be mutually disjoint.

Note that the low value interacts with the min domain uid option!

read only = yes|no

This option can be used to turn the writing backends tdb, tdb2, and ldap into read only mode. This can be useful e.g. in cases where a pre-filled database exists that should not be extended automatically.

The following example illustrates how to configure the idmap_ad(8) backend for the CORP domain and the idmap_tdb(8) backend for all other domains. This configuration assumes that the admin of CORP assigns unix ids below 1000000 via the SFU extensions, and winbind is supposed to use the next million entries for its own mappings from trusted domains and for local groups for example.

```
idmap config * : backend = tdb
idmap config * : range = 1000000-1999999
idmap config CORP : backend  = ad
idmap config CORP : range = 1000-999999
```

No default

winbind gid

This parameter is a synonym for idmap gid.

idmap gid (G)

The idmap gid parameter specifies the range of group ids for the default idmap configuration. It is now deprecated in favour of idmap config * : range.

See the idmap config option.

Default: idmap gid =

Example: idmap gid = 10000-20000

idmap negative cache time (G)

This parameter specifies the number of seconds that Winbind's idmap interface will cache negative SID/uid/gid query results.

Default: idmap negative cache time = 120

winbind uid

This parameter is a synonym for idmap uid.

idmap uid (G)

The idmap uid parameter specifies the range of user ids for the default idmap configuration. It is now deprecated in favour of idmap config * : range.

See the idmap config option.

Default: idmap uid =

Example: idmap uid = 10000-20000

include (S)

This allows you to include one config file inside another. The file is included literally, as though typed in place.

It takes the standard substitutions, except %u, %P and %S.

The parameter include = registry has a special meaning: It does not include a file named registry from the current working directory, but instead reads the global configuration options from the registry. See the section on registry-based configuration for details. Note that this option automatically activates registry shares.

Default: include =

Example: include = /usr/local/samba/lib/admin_smb.conf

include system krb5 conf (G)

Setting this parameter to no will prevent winbind to include the system /etc/krb5.conf file into the krb5.conf file it creates. See also create krb5 conf. This option only applies to Samba built with

MIT Kerberos.

Default: include system krb5 conf = yes

inherit acls (S)

This parameter can be used to ensure that if default acls exist on parent directories, they are always honored when creating a new file or subdirectory in these parent directories. The default behavior is to use the unix mode specified when creating the directory. Enabling this option sets the unix mode to 0777, thus guaranteeing that default directory acls are propagated. Note that using the VFS modules acl_xattr or acl_tdb which store native Windows as meta-data will automatically turn this option on for any share for which they are loaded, as they require this option to emulate Windows ACLs correctly.

Default: inherit acls = no

inherit owner (S)

The ownership of new files and directories is normally governed by effective uid of the connected user. This option allows the Samba administrator to specify that the ownership for new files and directories should be controlled by the ownership of the parent directory.

Valid options are:

? no - Both the Windows (SID) owner and the UNIX (uid) owner of the file are governed by the identity of the user that created the file.

? windows and unix - The Windows (SID) owner and the UNIX (uid) owner of new files and directories are set to the respective owner of the parent directory.

? yes - a synonym for windows and unix.

? unix only - Only the UNIX owner is set to the UNIX owner of the parent directory.

Common scenarios where this behavior is useful is in implementing drop-boxes, where users can create and edit files but not delete them and ensuring that newly created files in a user's roaming profile directory are actually owned by the user.

The unix only option effectively breaks the tie between the Windows owner of a file and the UNIX owner. As a logical consequence, in this mode, setting the the Windows owner of a file does not modify the UNIX owner. Using this mode should typically be combined with a backing store that can emulate the full NT ACL model without affecting the POSIX permissions, such as the acl_xattr VFS module, coupled with acl_xattr:ignore system acls = yes. This can be used to emulate folder quotas, when files are exposed only via SMB (without UNIX extensions). The UNIX owner of a directory is locally set and inherited by all subdirectories and files, and they all consume the same quota.

Default: inherit owner = no

inherit permissions (S)

The permissions on new files and directories are normally governed by create mask, directory mask, force create mode and force directory mode but the boolean inherit permissions parameter overrides this.

New directories inherit the mode of the parent directory, including bits such as setgid.

New files inherit their read/write bits from the parent directory.

Their execute bits continue to be determined by map archive, map hidden and map system as usual.

Note that the setuid bit is never set via inheritance (the code explicitly prohibits this).

This can be particularly useful on large systems with many users, perhaps several thousand, to allow a single [homes] share to be used flexibly by each user.

Default: inherit permissions = no

init logon delay (G)

This parameter specifies a delay in milliseconds for the hosts configured for delayed initial samlogon with init logon delayed hosts.

Default: init logon delay = 100

init logon delayed hosts (G)

This parameter takes a list of host names, addresses or networks for which the initial samlogon reply should be delayed (so other DCs get preferred by XP workstations if there are any).

The length of the delay can be specified with the init logon delay parameter.

Default: init logon delayed hosts =

Example: init logon delayed hosts = 150.203.5. myhost.mynet.de

interfaces (G)

This option allows you to override the default network interfaces list that Samba will use for browsing, name registration and other NetBIOS over TCP/IP (NBT) traffic. By default Samba will query the kernel for the list of all active interfaces and use any interfaces except 127.0.0.1 that are broadcast capable.

The option takes a list of interface strings. Each string can be in any of the following forms:

? a network interface name (such as eth0). This may include shell-like wildcards so eth* will match any interface starting with the substring "eth"

? an IP address. In this case the netmask is determined from the list of interfaces obtained from the kernel

? an IP/mask pair.

? a broadcast/mask pair.

The "mask" parameters can either be a bit length (such as 24 for a C class network) or a full netmask in dotted decimal form.

The "IP" parameters above can either be a full dotted decimal IP address or a hostname which will be looked up via the OS's normal hostname resolution mechanisms.

By default Samba enables all active interfaces that are broadcast capable except the loopback adaptor (IP address 127.0.0.1).

In order to support SMB3 multi-channel configurations, smbd understands some extra parameters which can be appended after the actual interface with this extended syntax (note that the quoting

is important in order to handle the ; and , characters):

"interface[;key1=value1[,key2=value2[...]]]"

Known keys are speed, capability, and if_index. Speed is specified

in bits per second. Known capabilities are RSS and RDMA. The

if_index should be used with care: the values must not coincide

with indexes used by the kernel. Note that these options are mainly

intended for testing and development rather than for production

use. At least on Linux systems, these values should be

auto-detected, but the settings can serve as last a resort when

autodetection is not working or is not available. The specified

values overwrite the auto-detected values.

The first two example below configures three network interfaces

corresponding to the eth0 device and IP addresses 192.168.2.10 and

192.168.3.10. The netmasks of the latter two interfaces would be

set to 255.255.255.0.

The other examples show how per interface extra parameters can be

specified. Notice the possible usage of "," and ";", which makes

the double quoting necessary.

Default: interfaces =

Example: interfaces = eth0 192.168.2.10/24

192.168.3.10/255.255.255.0

Example: interfaces = eth0, 192.168.2.10/24;

192.168.3.10/255.255.255.0

Example: interfaces =

"eth0;if_index=65,speed=1000000000,capability=RSS"

Example: interfaces = "lo;speed=1000000000" "eth0;capability=RSS"

Example: interfaces = "lo;speed=1000000000" , "eth0;capability=RSS"

Example: interfaces = "eth0;capability=RSS" ,

"rdma1;capability=RDMA" ; "rdma2;capability=RSS,capability=RDMA"

invalid users (S)

This is a list of users that should not be allowed to login to this

service. This is really a paranoid check to absolutely ensure an improper setting does not breach your security.

A name starting with a '@' is interpreted as an NIS netgroup first (if your system supports NIS), and then as a UNIX group if the name was not found in the NIS netgroup database.

A name starting with '+' is interpreted only by looking in the UNIX group database via the NSS getgrnam() interface. A name starting with '&' is interpreted only by looking in the NIS netgroup database (this requires NIS to be working on your system). The characters '+' and '&' may be used at the start of the name in either order so the value +&group means check the UNIX group database, followed by the NIS netgroup database, and the value &+group means check the NIS netgroup database, followed by the UNIX group database (the same as the '@' prefix).

The current servicename is substituted for %S. This is useful in the [homes] section.

Default: invalid users = # no invalid users

Example: invalid users = root fred admin @wheel

iprint server (G)

This parameter is only applicable if printing is set to iprint.

If set, this option overrides the ServerName option in the CUPS client.conf. This is necessary if you have virtual samba servers that connect to different CUPS daemons.

Default: iprint server = ""

Example: iprint server = MYCUPSSERVER

kdc default domain supported enctypes (G)

Set the default value of msDS-SupportedEncryptionTypes for service accounts in Active Directory that are missing this value or where msDS-SupportedEncryptionTypes is set to 0.

This allows Samba administrators to match the configuration flexibility provided by the HKEY_LOCAL_MACHINE\System\CurrentControlSet\services\KDC\DefaultDomainSupportedEncTypes Registry Value on Windows.

Unlike the Windows registry key (which only takes an base-10 number), in Samba this may also be expressed in hexadecimal or as a list of Kerberos encryption type names.

Specified values are ORed together bitwise, and those currently supported consist of:

? arcfour-hmac-md5, rc4-hmac, 0x4, or 4

Known on Windows as Kerberos RC4 encryption

? aes128-cts-hmac-sha1-96, aes128-cts, 0x8, or 8

Known on Windows as Kerberos AES 128 bit encryption

? aes256-cts-hmac-sha1-96, aes256-cts, 0x10, or 16

Known on Windows as Kerberos AES 256 bit encryption

? aes256-cts-hmac-sha1-96-sk, aes256-cts-sk, 0x20, or 32

Allow AES session keys. When this is set, it indicates to the KDC that AES session keys can be used, even when aes256-cts and aes128-cts are not set. This allows use of AES keys against hosts otherwise only configured with RC4 for ticket keys (which is the default).

Default: kdc default domain supported enctypes = 0 # maps to what the software supports currently: arcfour-hmac-md5 aes256-cts-hmac-sha1-96-sk

kdc enable fast (G)

With the Samba 4.16 the embedded Heimdal KDC brings support for RFC6113 FAST, which wasn't available in older Samba versions.

This option is mostly for testing and currently only applies if the embedded Heimdal KDC is used.

Default: kdc enable fast = yes

kdc force enable rc4 weak session keys (G)

RFC8429 declares that rc4-hmac Kerberos ciphers are weak and there are known attacks on Active Directory use of this cipher suite.

However for compatibility with Microsoft Windows this option allows the KDC to assume that regardless of the value set in a service account's msDS-SupportedEncryptionTypes attribute that a rc4-hmac Kerberos session key (as distinct from the ticket key, as found in a service keytab) can be used if the potentially older client requests it.

Default: kdc force enable rc4 weak session keys = no

kdc supported enctypes (G)

On an active directory domain controller, this is the list of supported encryption types for local running kdc.

This allows Samba administrators to remove support for weak/unused encryption types, similar the configuration flexibility provided by the Network security: Configure encryption types allowed for Kerberos GPO/Local Policies/Security Options Value, which results in the

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System\Kerberos\Parameters\SupportedEncryptionTypes

Registry Value on Windows.

Unlike the Windows registry key (which only takes an base-10 number), in Samba this may also be expressed as hexadecimal or a list of Kerberos encryption type names.

Specified values are ORed together bitwise, and those currently supported consist of:

? arcfour-hmac-md5, rc4-hmac, 0x4, or 4

Known on Windows as Kerberos RC4 encryption

? aes128-cts-hmac-sha1-96, aes128-cts, 0x8, or 8

Known on Windows as Kerberos AES 128 bit encryption

? aes256-cts-hmac-sha1-96, aes256-cts, 0x10, or 16

Known on Windows as Kerberos AES 256 bit encryption

Default: kdc supported enctypes = 0 # maps to what the software supports currently: arcfour-hmac-md5 aes128-cts-hmac-sha1-96 aes256-cts-hmac-sha1-96

keepalive (G)

The value of the parameter (an integer) represents the number of seconds between keepalive packets. If this parameter is zero, no keepalive packets will be sent. Keepalive packets, if sent, allow the server to tell whether a client is still present and responding.

Keepalives should, in general, not be needed if the socket has the SO_KEEPALIVE attribute set on it by default. (see socket options). Basically you should only use this option if you strike difficulties.

Please note this option only applies to SMB1 client connections, and has no effect on SMB2 clients.

Default: keepalive = 300

Example: keepalive = 600

kerberos encryption types (G)

This parameter determines the encryption types to use when operating as a Kerberos client. Possible values are all, strong, and legacy.

Samba uses a Kerberos library (MIT or Heimdal) to obtain Kerberos tickets. This library is normally configured outside of Samba, using the krb5.conf file. This file may also include directives to configure the encryption types to be used. However, Samba implements Active Directory protocols and algorithms to locate a domain controller. In order to force the Kerberos library into using the correct domain controller, some Samba processes, such as winbindd(8) and net(8), build a private krb5.conf file for use by the Kerberos library while being invoked from Samba. This private file controls all aspects of the Kerberos library operation, and this parameter controls how the encryption types are configured within this generated file, and therefore also controls the encryption types negotiable by Samba.

When set to all, all active directory encryption types are allowed.

When set to strong, only AES-based encryption types are offered. This can be used in hardened environments to prevent downgrade attacks.

When set to legacy, only RC4-HMAC-MD5 is allowed. AVOID using this option, because of CVE-2022-37966 see https://bugzilla.samba.org/show_bug.cgi?id=15237.

Default: kerberos encryption types = all

kerberos method (G)

Controls how kerberos tickets are verified.

Valid options are:

- ? secrets only - use only the secrets.tdb for ticket verification (default)
- ? system keytab - use only the system keytab for ticket verification
- ? dedicated keytab - use a dedicated keytab for ticket verification
- ? secrets and keytab - use the secrets.tdb first, then the system keytab

The major difference between "system keytab" and "dedicated keytab" is that the latter method relies on kerberos to find the correct keytab entry instead of filtering based on expected principals.

When the kerberos method is in "dedicated keytab" mode, dedicated keytab file must be set to specify the location of the keytab file.

Default: kerberos method = default

kernel change notify (G)

This parameter specifies whether Samba should ask the kernel for change notifications in directories so that SMB clients can refresh whenever the data on the server changes.

This parameter is only used when your kernel supports change notification to user programs using the inotify interface.

Default: kernel change notify = yes

kernel oplocks (S)

For UNIXes that support kernel based oplocks (currently only Linux), this parameter allows the use of them to be turned on or off. However, this disables Level II oplocks for clients as the Linux kernel does not support them properly.

Kernel oplocks support allows Samba oplocks to be broken whenever a local UNIX process or NFS operation accesses a file that smbd(8) has oplocked. This allows complete data consistency between

SMB/CIFS, NFS and local file access (and is a very cool feature :-).

If you do not need this interaction, you should disable the parameter on Linux to get Level II oplocks and the associated performance benefit.

This parameter defaults to no and is translated to a no-op on systems that do not have the necessary kernel support.

Default: kernel oplocks = no

kernel share modes (S)

This parameter controls whether SMB share modes are translated into file system specific sharemode calls.

Kernel share modes provide a minimal level of interoperability with local UNIX processes and NFS operations by preventing access corresponding to the SMB share modes. This requires a file system specific VFS module with proper support.

Note that in order to use SMB2 durable file handles on a share, you have to turn kernel share modes off.

This parameter defaults to no. Setting it to yes requires a file system module that supports file system sharemodes, otherwise attempts to access files will fail with a sharing violation.

Default: kernel share modes = no

kpasswd port (G)

Specifies which ports the Kerberos server should listen on for password changes.

Default: kpasswd port = 464

krb5 port (G)

Specifies which port the KDC should listen on for Kerberos traffic.

Default: krb5 port = 88

lanman auth (G)

This parameter has been deprecated since Samba 4.11 and support for LanMan (as distinct from NTLM, NTLMv2 or Kerberos authentication) will be removed in a future Samba release.

That is, in the future, the current default of lanman auth = no will be the enforced behaviour.

This parameter determines whether or not smbd(8) will attempt to authenticate users or permit password changes using the LANMAN password hash. If disabled, only clients which support NT password hashes (e.g. Windows NT/2000 clients, smbclient, but not Windows 95/98 or the MS DOS network client) will be able to connect to the Samba host.

The LANMAN encrypted response is easily broken, due to its case-insensitive nature, and the choice of algorithm. Servers without Windows 95/98/ME or MS DOS clients are advised to disable this option.

When this parameter is set to no this will also result in sambaLMPassword in Samba's passdb being blanked after the next password change. As a result of that lanman clients won't be able to authenticate, even if lanman auth is re-enabled later on.

Unlike the encrypt passwords option, this parameter cannot alter client behaviour, and the LANMAN response will still be sent over the network. See the client lanman auth to disable this for Samba's clients (such as smbclient)

This parameter is overridden by ntlm auth, so unless that it is also set to ntlmv1-permitted or yes, then only NTLMv2 logins will be permitted and no LM hash will be stored. All modern clients support NTLMv2, and but some older clients require special configuration to use it.

This parameter has no impact on the Samba AD DC, LM authentication is always disabled and no LM password is ever stored.

Default: lanman auth = no

large readwrite (G)

This parameter determines whether or not smbd(8) supports the new 64k streaming read and write variant SMB requests introduced with Windows 2000. Note that due to Windows 2000 client redirector bugs this requires Samba to be running on a 64-bit capable operating system such as IRIX, Solaris or a Linux 2.4 kernel. Can improve performance by 10% with Windows 2000 clients. Defaults to on. Not as tested as some other Samba code paths.

Default: large readwrite = yes

ldap admin dn (G)

The ldap admin dn defines the Distinguished Name (DN) name used by Samba to contact the ldap server when retrieving user account information. The ldap admin dn is used in conjunction with the admin dn password stored in the private/secrets.tdb file. See the smbpasswd(8) man page for more information on how to accomplish this.

The ldap admin dn requires a fully specified DN. The ldap suffix is not appended to the ldap admin dn.

No default

ldap connection timeout (G)

This parameter tells the LDAP library calls which timeout in seconds they should honor during initial connection establishments to LDAP servers. It is very useful in failover scenarios in particular. If one or more LDAP servers are not reachable at all, we do not have to wait until TCP timeouts are over. This feature must be supported by your LDAP library.

This parameter is different from ldap timeout which affects operations on LDAP servers using an existing connection and not establishing an initial connection.

Default: ldap connection timeout = 2

ldap debug level (G)

This parameter controls the debug level of the LDAP library calls. In the case of OpenLDAP, it is the same bit-field as understood by the server and documented in the slapd.conf(5) manpage. A typical useful value will be 1 for tracing function calls.

The debug output from the LDAP libraries appears with the prefix [LDAP] in Samba's logging output. The level at which LDAP logging is printed is controlled by the parameter ldap debug threshold.

Default: ldap debug level = 0

Example: ldap debug level = 1

ldap debug threshold (G)

This parameter controls the Samba debug level at which the ldap library debug output is printed in the Samba logs. See the description of ldap debug level for details.

Default: ldap debug threshold = 10

Example: ldap debug threshold = 5

ldap delete dn (G)

This parameter specifies whether a delete operation in the ldapsam deletes the complete entry or only the attributes specific to Samba.

Default: ldap delete dn = no

ldap deref (G)

This option controls whether Samba should tell the LDAP library to use a certain alias dereferencing method. The default is auto, which means that the default setting of the ldap client library will be kept. Other possible values are never, finding, searching and always. Grab your LDAP manual for more information.

Default: ldap deref = auto

Example: ldap deref = searching

ldap follow referral (G)

This option controls whether to follow LDAP referrals or not when searching for entries in the LDAP database. Possible values are on to enable following referrals, off to disable this, and auto, to use the libldap default settings. libldap's choice of following referrals or not is set in /etc/openldap/ldap.conf with the REFERRALS parameter as documented in ldap.conf(5).

Default: ldap follow referral = auto

Example: ldap follow referral = off

ldap group suffix (G)

This parameter specifies the suffix that is used for groups when these are added to the LDAP directory. If this parameter is unset, the value of ldap suffix will be used instead. The suffix string is pre-pended to the ldap suffix string so use a partial DN.

Default: ldap group suffix =

Example: ldap group suffix = ou=Groups

ldap idmap suffix (G)

This parameters specifies the suffix that is used when storing idmap mappings. If this parameter is unset, the value of ldap suffix will be used instead. The suffix string is pre-pended to the ldap suffix string so use a partial DN.

Default: ldap idmap suffix =

Example: ldap idmap suffix = ou=Idmap

ldap machine suffix (G)

It specifies where machines should be added to the ldap tree. If this parameter is unset, the value of ldap suffix will be used instead. The suffix string is pre-pended to the ldap suffix string so use a partial DN.

Default: ldap machine suffix =

Example: ldap machine suffix = ou=Computers

ldap max anonymous request size (G)

This parameter specifies the maximum permitted size (in bytes) for an LDAP request received on an anonymous connection.

If the request size exceeds this limit the request will be rejected.

Default: ldap max anonymous request size = 256000

Example: ldap max anonymous request size = 500000

ldap max authenticated request size (G)

This parameter specifies the maximum permitted size (in bytes) for an LDAP request received on an authenticated connection.

If the request size exceeds this limit the request will be rejected.

Default: ldap max authenticated request size = 16777216

Example: ldap max authenticated request size = 4194304

ldap max search request size (G)

This parameter specifies the maximum permitted size (in bytes) for an LDAP search request.

If the request size exceeds this limit the request will be rejected.

Default: ldap max search request size = 256000

Example: ldap max search request size = 4194304

ldap page size (G)

This parameter specifies the number of entries per page.

If the LDAP server supports paged results, clients can request subsets of search results (pages) instead of the entire list. This parameter specifies the size of these pages.

Default: ldap page size = 1000

Example: ldap page size = 512

ldap password sync

This parameter is a synonym for ldap passwd sync.

ldap passwd sync (G)

This option is used to define whether or not Samba should sync the LDAP password with the NT and LM hashes for normal accounts (NOT for workstation, server or domain trusts) on a password change via SAMBA.

The ldap passwd sync can be set to one of three values:

? Yes = Try to update the LDAP, NT and LM passwords and update the pwdLastSet time.

? No = Update NT and LM passwords and update the pwdLastSet time.

? Only = Only update the LDAP password and let the LDAP server do the rest.

Default: ldap passwd sync = no

ldap replication sleep (G)

When Samba is asked to write to a read-only LDAP replica, we are redirected to talk to the read-write master server. This server then replicates our changes back to the 'local' server, however the replication might take some seconds, especially over slow links.

Certain client activities, particularly domain joins, can become confused by the 'success' that does not immediately change the LDAP back-end's data.

This option simply causes Samba to wait a short time, to allow the LDAP server to catch up. If you have a particularly high-latency network, you may wish to time the LDAP replication with a network sniffer, and increase this value accordingly. Be aware that no checking is performed that the data has actually replicated.

The value is specified in milliseconds, the maximum value is 5000 (5 seconds).

Default: ldap replication sleep = 1000

ldapsam:editposix (G)

Editposix is an option that leverages ldapsam:trusted to make it simpler to manage a domain controller eliminating the need to set up custom scripts to add and manage the posix users and groups. This option will instead directly manipulate the ldap tree to create, remove and modify user and group entries. This option also requires a running winbindd as it is used to allocate new uids/gids on user/group creation. The allocation range must be therefore configured.

To use this option, a basic ldap tree must be provided and the ldap suffix parameters must be properly configured. On virgin servers the default users and groups (Administrator, Guest, Domain Users, Domain Admins, Domain Guests) can be precreated with the command net sam provision. To run this command the ldap server must be running, Winbindd must be running and the smb.conf ldap options must be properly configured. The typical ldap setup used with the ldapsam:trusted = yes option is usually sufficient to use ldapsam:editposix = yes as well.

An example configuration can be the following:

```
encrypt passwords = true
passdb backend = ldapsam
ldapsam:trusted=yes
ldapsam:editposix=yes
```

```
ldap admin dn = cn=admin,dc=samba,dc=org
ldap delete dn = yes
ldap group suffix = ou=groups
ldap idmap suffix = ou=idmap
ldap machine suffix = ou=computers
ldap user suffix = ou=users
ldap suffix = dc=samba,dc=org
idmap backend = ldap:"ldap://localhost"
idmap uid = 5000-50000
idmap gid = 5000-50000
```

This configuration assumes a directory layout like described in the following ldif:

```
dn: dc=samba,dc=org
objectClass: top
objectClass: dcObject
objectClass: organization
o: samba.org
dc: samba
dn: cn=admin,dc=samba,dc=org
objectClass: simpleSecurityObject
objectClass: organizationalRole
cn: admin
description: LDAP administrator
userPassword: secret
dn: ou=users,dc=samba,dc=org
objectClass: top
objectClass: organizationalUnit
ou: users
dn: ou=groups,dc=samba,dc=org
objectClass: top
objectClass: organizationalUnit
ou: groups
dn: ou=idmap,dc=samba,dc=org
```

```
objectClass: top
objectClass: organizationalUnit
ou: idmap
dn: ou=computers,dc=samba,dc=org
objectClass: top
objectClass: organizationalUnit
ou: computers
```

Default: ldapsam:editposix = no

ldapsam:trusted (G)

By default, Samba as a Domain Controller with an LDAP backend needs to use the Unix-style NSS subsystem to access user and group information. Due to the way Unix stores user information in /etc/passwd and /etc/group this inevitably leads to inefficiencies. One important question a user needs to know is the list of groups he is member of. The plain UNIX model involves a complete enumeration of the file /etc/group and its NSS counterparts in LDAP. UNIX has optimized functions to enumerate group membership. Sadly, other functions that are used to deal with user and group attributes lack such optimization.

To make Samba scale well in large environments, the ldapsam:trusted = yes option assumes that the complete user and group database that is relevant to Samba is stored in LDAP with the standard posixAccount/posixGroup attributes. It further assumes that the Samba auxiliary object classes are stored together with the POSIX data in the same LDAP object. If these assumptions are met, ldapsam:trusted = yes can be activated and Samba can bypass the NSS system to query user group memberships. Optimized LDAP queries can greatly speed up domain logon and administration tasks. Depending on the size of the LDAP database a factor of 100 or more for common queries is easily achieved.

Default: ldapsam:trusted = no

ldap server require strong auth (G)

The ldap server require strong auth defines whether the ldap server requires ldap traffic to be signed or signed and encrypted (sealed). Possible values are no, allow_sasl_over_tls and yes.

A value of no allows simple and sasl binds over all transports.

A value of allow_sasl_over_tls allows simple and sasl binds (without sign or seal) over TLS encrypted connections. Unencrypted connections only allow sasl binds with sign or seal.

A value of yes allows only simple binds over TLS encrypted connections. Unencrypted connections only allow sasl binds with sign or seal.

Default: ldap server require strong auth = yes

ldap ssl (G)

This option is used to define whether or not Samba should use SSL when connecting to the ldap server This is NOT related to Samba's previous SSL support which was enabled by specifying the --with-ssl option to the configure script.

LDAP connections should be secured where possible. This may be done setting either this parameter to start tls or by specifying ldaps:// in the URL argument of passdb backend.

The ldap ssl can be set to one of two values:

? Off = Never use SSL when querying the directory.

? start tls = Use the LDAPv3 StartTLS extended operation (RFC2830) for communicating with the directory server.

Please note that this parameter does only affect rpc methods.

Default: ldap ssl = start tls

ldap suffix (G)

Specifies the base for all ldap suffixes and for storing the sambaDomain object.

The ldap suffix will be appended to the values specified for the ldap user suffix, ldap group suffix, ldap machine suffix, and the ldap idmap suffix. Each of these should be given only a DN relative to the ldap suffix.

Default: ldap suffix =

Example: ldap suffix = dc=samba,dc=org

ldap timeout (G)

This parameter defines the number of seconds that Samba should use as timeout for LDAP operations.

Default: ldap timeout = 15

ldap user suffix (G)

This parameter specifies where users are added to the tree. If this parameter is unset, the value of ldap suffix will be used instead.

The suffix string is pre-pended to the ldap suffix string so use a partial DN.

Default: ldap user suffix =

Example: ldap user suffix = ou=people

level2 oplocks (S)

This parameter controls whether Samba supports level2 (read-only) oplocks on a share.

Level2, or read-only oplocks allow Windows NT clients that have an oplock on a file to downgrade from a read-write oplock to a read-only oplock once a second client opens the file (instead of releasing all oplocks on a second open, as in traditional, exclusive oplocks). This allows all openers of the file that support level2 oplocks to cache the file for read-ahead only (ie. they may not cache writes or lock requests) and increases performance for many accesses of files that are not commonly written (such as application .EXE files).

Once one of the clients which have a read-only oplock writes to the file all clients are notified (no reply is needed or waited for) and told to break their oplocks to "none" and delete any read-ahead caches.

It is recommended that this parameter be turned on to speed access to shared executables.

For more discussions on level2 oplocks see the CIFS spec.

Currently, if kernel oplocks are supported then level2 oplocks are not granted (even if this parameter is set to yes). Note also, the oplocks parameter must be set to yes on this share in order for this parameter to have any effect.

Default: level2 oplocks = yes

lm announce (G)

This parameter determines if nmbd(8) will produce Lanman announce broadcasts that are needed by OS/2 clients in order for them to see the Samba server in their browse list. This parameter can have three values, yes, no, or auto. The default is auto. If set to no Samba will never produce these broadcasts. If set to yes Samba will produce Lanman announce broadcasts at a frequency set by the parameter lm interval. If set to auto Samba will not send Lanman announce broadcasts by default but will listen for them. If it hears such a broadcast on the wire it will then start sending them at a frequency set by the parameter lm interval.

Default: lm announce = auto

Example: lm announce = yes

lm interval (G)

If Samba is set to produce Lanman announce broadcasts needed by OS/2 clients (see the lm announce parameter) then this parameter defines the frequency in seconds with which they will be made. If this is set to zero then no Lanman announcements will be made despite the setting of the lm announce parameter.

Default: lm interval = 60

Example: lm interval = 120

load printers (G)

A boolean variable that controls whether all printers in the printcap will be loaded for browsing by default. See the printers section for more details.

Default: load printers = yes

local master (G)

This option allows nmbd(8) to try and become a local master browser on a subnet. If set to no then nmbd will not attempt to become a local master browser on a subnet and will also lose in all browsing elections. By default this value is set to yes. Setting this value to yes doesn't mean that Samba will become the local master browser on a subnet, just that nmbd will participate in elections for local master browser.

Setting this value to no will cause nmbd never to become a local master browser.

Default: local master = yes

lock dir

This parameter is a synonym for lock directory.

lock directory (G)

This option specifies the directory where lock files will be placed. The lock files are used to implement the max connections option.

Note: This option can not be set inside registry configurations.

The files placed in this directory are not required across service restarts and can be safely placed on volatile storage (e.g. tmpfs in Linux)

Default: lock directory = /var/lib/samba/lock

Example: lock directory = /var/run/samba/locks

locking (S)

This controls whether or not locking will be performed by the server in response to lock requests from the client.

If locking = no, all lock and unlock requests will appear to succeed and all lock queries will report that the file in question is available for locking.

If locking = yes, real locking will be performed by the server.

This option may be useful for read-only filesystems which may not need locking (such as CDROM drives), although setting this parameter of no is not really recommended even in this case.

Be careful about disabling locking either globally or in a specific service, as lack of locking may result in data corruption. You should never need to set this parameter.

Default: locking = yes

lock spin time (G)

The time in milliseconds that smbd should keep waiting to see if a failed lock request can be granted. This parameter has changed in default value from Samba 3.0.23 from 10 to 200. The associated lock spin count parameter is no longer used in Samba 3.0.24. You should not need to change the value of this parameter.

Default: lock spin time = 200

log file (G)

This option allows you to override the name of the Samba log file (also known as the debug file).

This option takes the standard substitutions, allowing you to have separate log files for each user or machine.

No default

Example: log file = /usr/local/samba/var/log.%m

logging (G)

This parameter configures logging backends. Multiple backends can be specified at the same time, with different log levels for each backend. The parameter is a list of backends, where each backend is specified as backend[:option][@loglevel].

The 'option' parameter can be used to pass backend-specific options.

The log level for a backend is optional, if it is not set for a backend, all messages are sent to this backend. The parameter log level determines overall log levels, while the log levels specified here define what is sent to the individual backends.

When logging is set, it overrides the syslog and syslog only parameters.

Some backends are only available when Samba has been compiled with the additional libraries. The overall list of logging backends:

? syslog

? file

? systemd

? lttng

? gpfs

? ringbuf

The ringbuf backend supports an optional size argument to change the buffer size used, the default is 1 MB: ringbuf:size=NBYTES

Default: logging =

Example: logging = syslog@1 file

debuglevel

This parameter is a synonym for log level.

log level (G)

The value of the parameter (a string) allows the debug level (logging level) to be specified in the smb.conf file.

This parameter has been extended since the 2.2.x series, now it allows one to specify the debug level for multiple debug classes and distinct logfiles for debug classes. This is to give greater flexibility in the configuration of the system. The following debug classes are currently implemented:

? all

? tdb

? printdrivers

? lanman

? smb

? rpc_parse

? rpc_srv

? rpc_cli

? passdb

? sam

? auth

? winbind

? vfs

? idmap

? quota

? acls

? locking

? msdfs

? dmapi

? registry

? scavenger

? dns

? ldb

? tevent

? auth_audit

? auth_json_audit

? kerberos

? drs_repl

? smb2

? smb2_credits

? dsdb_audit

? dsdb_json_audit

? dsdb_password_audit

? dsdb_password_json_audit

? dsdb_transaction_audit

? dsdb_transaction_json_audit

? dsdb_group_audit

? dsdb_group_json_audit

Various modules register dynamic debug classes at first usage:

? catia

? dfs_samba4

? extd_audit

? fileid

? fruit

? full_audit

? media_harmony

? preopen

? recycle

? shadow_copy

? shadow_copy

? unityed_media

? virusfilter

To configure the logging for specific classes to go into a different file then log file, you can append @PATH to the class, eg log level = 1 full_audit:1@/var/log/audit.log.

Authentication and authorization audit information is logged under the auth_audit, and if Samba was not compiled with --without-json, a JSON representation is logged under auth_json_audit.

Support is comprehensive for all authentication and authorisation of user accounts in the Samba Active Directory Domain Controller, as well as the implicit authentication in password changes. In the file server, NTLM authentication, SMB and RPC authorization is covered.

Log levels for auth_audit and auth_audit_json are:

? 2: Authentication Failure

? 3: Authentication Success

? 4: Authorization Success

? 5: Anonymous Authentication and Authorization Success

Changes to the AD DC sam.ldb database are logged under the dsdb_audit and a JSON representation is logged under dsdb_json_audit.

Group membership changes to the AD DC sam.ldb database are logged under the dsdb_group_audit and a JSON representation is logged under dsdb_group_json_audit.

Log levels for dsdb_audit, dsdb_json_audit, dsdb_group_audit, dsdb_group_json_audit and dsdb_json_audit are:

? 5: Database modifications

? 5: Replicated updates from another DC

Password changes and Password resets in the AD DC are logged under dsdb_password_audit and a JSON representation is logged under the dsdb_password_json_audit. Password changes will also appears as authentication events via auth_audit and auth_audit_json.

Log levels for dsdb_password_audit and dsdb_password_json_audit are:

- ? 5: Successful password changes and resets

Transaction rollbacks and prepare commit failures are logged under the dsdb_transaction_audit and a JSON representation is logged under the dsdb_transaction_json_audit.

Log levels for dsdb_transaction_audit and dsdb_transaction_json are:

- ? 5: Transaction failure (rollback)
- ? 10: Transaction success (commit)

Transaction roll-backs are possible in Samba, and whilst they rarely reflect anything more than the failure of an individual operation (say due to the add of a conflicting record), they are possible. Audit logs are already generated and sent to the system logs before the transaction is complete. Logging the transaction details allows the identification of password and sam.ldb operations that have been rolled back, and so have not actually persisted.

> Warning
>
> Changes to sam.ldb made locally by the root user with direct access to the database are not logged to the system logs, but to the administrator's own console. While less than ideal, any user able to make such modifications could disable the audit logging in any case.

Default: log level = 0

Example: log level = 3 passdb:5 auth:10 winbind:2

Example: log level = 1 full_audit:1@/var/log/audit.log winbind:2

log nt token command (G)

This option can be set to a command that will be called when new nt tokens are created.

This is only useful for development purposes.

Default: log nt token command =

logon drive (G)

This parameter specifies the local path to which the home directory will be connected (see logon home) and is only used by NT

Workstations.

Note that this option is only useful if Samba is set up as a logon server.

Default: logon drive =

Example: logon drive = h:

logon home (G)

This parameter specifies the home directory location when a Win95/98 or NT Workstation logs into a Samba PDC. It allows you to do

C:\>NET USE H: /HOME

from a command prompt, for example.

This option takes the standard substitutions, allowing you to have separate logon scripts for each user or machine.

This parameter can be used with Win9X workstations to ensure that roaming profiles are stored in a subdirectory of the user's home directory. This is done in the following way:

logon home = \\%N\%U\profile

This tells Samba to return the above string, with substitutions made when a client requests the info, generally in a NetUserGetInfo request. Win9X clients truncate the info to \\server\share when a user does net use /home but use the whole string when dealing with profiles.

Note that in prior versions of Samba, the logon path was returned rather than logon home. This broke net use /home but allowed profiles outside the home directory. The current implementation is correct, and can be used for profiles if you use the above trick.

Disable this feature by setting logon home = "" - using the empty string.

This option is only useful if Samba is set up as a logon server.

Default: logon home = \\%N\%U

Example: logon home = \\remote_smb_server\%U

logon path (G)

This parameter specifies the directory where roaming profiles

(Desktop, NTuser.dat, etc) are stored. Contrary to previous versions of these manual pages, it has nothing to do with Win 9X roaming profiles. To find out how to handle roaming profiles for Win 9X system, see the logon home parameter.

This option takes the standard substitutions, allowing you to have separate logon scripts for each user or machine. It also specifies the directory from which the "Application Data", desktop, start menu, network neighborhood, programs and other folders, and their contents, are loaded and displayed on your Windows NT client.

The share and the path must be readable by the user for the preferences and directories to be loaded onto the Windows NT client. The share must be writeable when the user logs in for the first time, in order that the Windows NT client can create the NTuser.dat and other directories. Thereafter, the directories and any of the contents can, if required, be made read-only. It is not advisable that the NTuser.dat file be made read-only - rename it to NTuser.man to achieve the desired effect (a MANdatory profile).

Windows clients can sometimes maintain a connection to the [homes] share, even though there is no user logged in. Therefore, it is vital that the logon path does not include a reference to the homes share (i.e. setting this parameter to \\%N\homes\profile_path will cause problems).

This option takes the standard substitutions, allowing you to have separate logon scripts for each user or machine.

Warning

Do not quote the value. Setting this as ?\\%N\profile\%U? will break profile handling. Where the tdbsam or ldapsam passdb backend is used, at the time the user account is created the value configured for this parameter is written to the passdb backend and that value will over-ride the parameter value present in the smb.conf file. Any error present in the passdb backend account record must be editted using the appropriate tool (pdbedit on the command-line, or any other locally provided system tool).

Note that this option is only useful if Samba is set up as a domain controller.

Disable the use of roaming profiles by setting the value of this parameter to the empty string. For example, logon path = "". Take note that even if the default setting in the smb.conf file is the empty string, any value specified in the user account settings in the passdb backend will over-ride the effect of setting this parameter to null. Disabling of all roaming profile use requires that the user account settings must also be blank.

An example of use is:

```
logon path = \\PROFILESERVER\PROFILE\%U
```

Default: logon path = \\%N\%U\profile

logon script (G)

This parameter specifies the batch file (.bat) or NT command file (.cmd) to be downloaded and run on a machine when a user successfully logs in. The file must contain the DOS style CR/LF line endings. Using a DOS-style editor to create the file is recommended.

The script must be a relative path to the [netlogon] service. If the [netlogon] service specifies a path of /usr/local/samba/netlogon, and logon script = STARTUP.BAT, then the file that will be downloaded is:

```
/usr/local/samba/netlogon/STARTUP.BAT
```

The contents of the batch file are entirely your choice. A suggested command would be to add NET TIME \\SERVER /SET /YES, to force every machine to synchronize clocks with the same time server. Another use would be to add NET USE U: \\SERVER\UTILS for commonly used utilities, or

```
NET USE Q: \\SERVER\ISO9001_QA
```

for example.

Note that it is particularly important not to allow write access to the [netlogon] share, or to grant users write permission on the batch files in a secure environment, as this would allow the batch files to be arbitrarily modified and security to be breached.

This option takes the standard substitutions, allowing you to have separate logon scripts for each user or machine.

This option is only useful if Samba is set up as a logon server in a classic domain controller role. If Samba is set up as an Active Directory domain controller, LDAP attribute scriptPath is used instead. For configurations where passdb backend = ldapsam is in use, this option only defines a default value in case LDAP attribute sambaLogonScript is missing.

Default: logon script =

Example: logon script = scripts\%U.bat

log writeable files on exit (G)

When the network connection between a CIFS client and Samba dies, Samba has no option but to simply shut down the server side of the network connection. If this happens, there is a risk of data corruption because the Windows client did not complete all write operations that the Windows application requested. Setting this option to "yes" makes smbd log with a level 0 message a list of all files that have been opened for writing when the network connection died. Those are the files that are potentially corrupted. It is meant as an aid for the administrator to give him a list of files to do consistency checks on.

Default: log writeable files on exit = no

lppause command (S)

This parameter specifies the command to be executed on the server host in order to stop printing or spooling a specific print job.

This command should be a program or script which takes a printer name and job number to pause the print job. One way of implementing this is by using job priorities, where jobs having a too low priority won't be sent to the printer.

If a %p is given then the printer name is put in its place. A %j is replaced with the job number (an integer). On HPUX (see printing=hpux), if the -p%p option is added to the lpq command, the job will show up with the correct status, i.e. if the job priority is lower than the set fence priority it will have the PAUSED status, whereas if the priority is equal or higher it will have the SPOOLED or PRINTING status.

Note that it is good practice to include the absolute path in the lppause command as the PATH may not be available to the server.

Currently no default value is given to this string, unless the value of the printing parameter is SYSV, in which case the default is : lp -i %p-%j -H hold or if the value of the printing parameter is SOFTQ, then the default is: qstat -s -j%j -h.

Default: lppause command = # determined by printing parameter

Example: lppause command = /usr/bin/lpalt %p-%j -p0

lpq cache time (G)

This controls how long lpq info will be cached for to prevent the lpq command being called too often. A separate cache is kept for each variation of the lpq command used by the system, so if you use different lpq commands for different users then they won't share cache information.

The cache files are stored in /tmp/lpq.xxxx where xxxx is a hash of the lpq command in use.

The default is 30 seconds, meaning that the cached results of a previous identical lpq command will be used if the cached data is less than 30 seconds old. A large value may be advisable if your lpq command is very slow.

A value of 0 will disable caching completely.

Default: lpq cache time = 30

Example: lpq cache time = 10

lpq command (S)

This parameter specifies the command to be executed on the server host in order to obtain lpq-style printer status information.

This command should be a program or script which takes a printer name as its only parameter and outputs printer status information.

Currently nine styles of printer status information are supported; BSD, AIX, LPRNG, PLP, SYSV, HPUX, QNX, CUPS, and SOFTQ. This covers most UNIX systems. You control which type is expected using the printing = option.

Some clients (notably Windows for Workgroups) may not correctly send the connection number for the printer they are requesting status information about. To get around this, the server reports on the first printer service connected to by the client. This only happens if the connection number sent is invalid.

If a %p is given then the printer name is put in its place. Otherwise it is placed at the end of the command.

Note that it is good practice to include the absolute path in the lpq command as the $PATH may not be available to the server. When compiled with the CUPS libraries, no lpq command is needed because smbd will make a library call to obtain the print queue listing.

Default: lpq command = # determined by printing parameter

Example: lpq command = /usr/bin/lpq -P%p

lpresume command (S)

This parameter specifies the command to be executed on the server host in order to restart or continue printing or spooling a specific print job.

This command should be a program or script which takes a printer name and job number to resume the print job. See also the lppause command parameter.

If a %p is given then the printer name is put in its place. A %j is replaced with the job number (an integer).

Note that it is good practice to include the absolute path in the lpresume command as the PATH may not be available to the server.

See also the printing parameter.

Default: Currently no default value is given to this string, unless the value of the printing parameter is SYSV, in which case the default is:

lp -i %p-%j -H resume

or if the value of the printing parameter is SOFTQ, then the default is:

qstat -s -j%j -r

Default: lpresume command = # determined by printing parameter

Example: lpresume command = /usr/bin/lpalt %p-%j -p2

lprm command (S)

This parameter specifies the command to be executed on the server host in order to delete a print job.

This command should be a program or script which takes a printer name and job number, and deletes the print job.

If a %p is given then the printer name is put in its place. A %j is replaced with the job number (an integer).

Note that it is good practice to include the absolute path in the lprm command as the PATH may not be available to the server.

Examples of use are:

lprm command = /usr/bin/lprm -P%p %j

or

lprm command = /usr/bin/cancel %p-%j

Default: lprm command = # determined by printing parameter

lsa over netlogon (G)

Setting this deprecated option will allow the RPC server in the AD DC to answer the LSARPC interface on the \pipe\netlogon IPC pipe.

When enabled, this matches the behaviour of Microsoft's Windows, due to their internal implementation choices.

If it is disabled (the default), the AD DC can offer improved performance, as the netlogon server is decoupled and can run as multiple processes.

Default: lsa over netlogon = no

machine password timeout (G)

If a Samba server is a member of a Windows NT or Active Directory Domain (see the security = domain and security = ads parameters), then periodically a running winbindd process will try and change the MACHINE ACCOUNT PASSWORD stored in the TDB called secrets.tdb.

This parameter specifies how often this password will be changed, in seconds. The default is one week (expressed in seconds), the same as a Windows NT Domain member server.

See also smbpasswd(8), and the security = domain and security = ads parameters.

Default: machine password timeout = 604800

magic output (S)

This parameter specifies the name of a file which will contain output created by a magic script (see the magic script parameter below).

Warning

If two clients use the same magic script in the same directory the output file content is undefined.

Default: magic output = # <magic script name>.out

Example: magic output = myfile.txt

magic script (S)

This parameter specifies the name of a file which, if opened, will be executed by the server when the file is closed. This allows a UNIX script to be sent to the Samba host and executed on behalf of the connected user.

Scripts executed in this way will be deleted upon completion assuming that the user has the appropriate level of privilege and the file permissions allow the deletion.

If the script generates output, output will be sent to the file specified by the magic output parameter (see above).

Note that some shells are unable to interpret scripts containing CR/LF instead of CR as the end-of-line marker. Magic scripts must be executable as is on the host, which for some hosts and some shells will require filtering at the DOS end.

Magic scripts are EXPERIMENTAL and should NOT be relied upon.

Default: magic script =

Example: magic script = user.csh

mangled names (S)

This controls whether non-DOS names under UNIX should be mapped to DOS-compatible names ("mangled") and made visible, or whether non-DOS names should simply be ignored.

See the section on name mangling for details on how to control the mangling process.

Possible option settings are

? yes - enables name mangling for all not DOS 8.3 conforming names.

? no - disables any name mangling.

? illegal (default) - does mangling for names with illegal NTFS characters. This is the most sensible setting for modern clients that don't use the shortname anymore.

If mangling is used then the mangling method is as follows:

? The first (up to) five alphanumeric characters before the rightmost dot of the filename are preserved, forced to upper case, and appear as the first (up to) five characters of the mangled name.

? A tilde "~" is appended to the first part of the mangled name, followed by a two-character unique sequence, based on the original root name (i.e., the original filename minus its final extension). The final extension is included in the hash calculation only if it contains any upper case characters or is longer than three characters.

Note that the character to use may be specified using the mangling char option, if you don't like '~'.

? Files whose UNIX name begins with a dot will be presented as DOS hidden files. The mangled name will be created as for other filenames, but with the leading dot removed and "___" as its extension regardless of actual original extension (that's three underscores).

The two-digit hash value consists of upper case alphanumeric characters.

This algorithm can cause name collisions only if files in a
directory share the same first five alphanumeric characters. The
probability of such a clash is 1/1300.

The name mangling (if enabled) allows a file to be copied between
UNIX directories from Windows/DOS while retaining the long UNIX
filename. UNIX files can be renamed to a new extension from
Windows/DOS and will retain the same basename. Mangled names do not
change between sessions.

Default: mangled names = illegal

Example: mangled names = no

mangle prefix (G)

controls the number of prefix characters from the original name
used when generating the mangled names. A larger value will give a
weaker hash and therefore more name collisions. The minimum value
is 1 and the maximum value is 6.

mangle prefix is effective only when mangling method is hash2.

Default: mangle prefix = 1

Example: mangle prefix = 4

mangling char (S)

This controls what character is used as the magic character in name
mangling. The default is a '~' but this may interfere with some
software. Use this option to set it to whatever you prefer. This is
effective only when mangling method is hash.

Default: mangling char = ~

Example: mangling char = ^

mangling method (G)

controls the algorithm used for the generating the mangled names.
Can take two different values, "hash" and "hash2". "hash" is the
algorithm that was used in Samba for many years and was the default
in Samba 2.2.x "hash2" is now the default and is newer and
considered a better algorithm (generates less collisions) in the
names. Many Win32 applications store the mangled names and so
changing to algorithms must not be done lightly as these

applications may break unless reinstalled.

Default: mangling method = hash2

Example: mangling method = hash

map acl inherit (S)

This boolean parameter controls whether smbd(8) will attempt to map the 'inherit' and 'protected' access control entry flags stored in Windows ACLs into an extended attribute called user.SAMBA_PAI (POSIX ACL Inheritance). This parameter requires supports for extended attributes on the filesystem and allows the Windows ACL editor to store inheritance information while NT ACLs are mapped best-effort to the POSIX ACLs.

Default: map acl inherit = no

map archive (S)

This controls whether the DOS archive attribute should be mapped to the UNIX owner execute bit. The DOS archive bit is set when a file has been modified since its last backup. One motivation for this option is to keep Samba/your PC from making any file it touches from becoming executable under UNIX. This can be quite annoying for shared source code, documents, etc...

Note that this parameter will be ignored if the store dos attributes parameter is set, as the DOS archive attribute will then be stored inside a UNIX extended attribute.

Note that this requires the create mask parameter to be set such that owner execute bit is not masked out (i.e. it must include 100). See the parameter create mask for details.

Default: map archive = yes

map hidden (S)

This controls whether DOS style hidden files should be mapped to the UNIX world execute bit.

Note that this parameter will be ignored if the store dos attributes parameter is set, as the DOS hidden attribute will then be stored inside a UNIX extended attribute.

Note that this requires the create mask to be set such that the world execute bit is not masked out (i.e. it must include 001). See the parameter create mask for details.

Default: map hidden = no

map readonly (S)

This controls how the DOS read only attribute should be mapped from a UNIX filesystem.

This parameter can take three different values, which tell smbd(8) how to display the read only attribute on files, where either store dos attributes is set to No, or no extended attribute is present. If store dos attributes is set to yes then this parameter is ignored. This is a new parameter introduced in Samba version 3.0.21.

The three settings are :

? Yes - The read only DOS attribute is mapped to the inverse of the user or owner write bit in the unix permission mode set. If the owner write bit is not set, the read only attribute is reported as being set on the file. If the read only DOS attribute is set, Samba sets the owner, group and others write bits to zero. Write bits set in an ACL are ignored by Samba. If the read only DOS attribute is unset, Samba simply sets the write bit of the owner to one.

? Permissions - The read only DOS attribute is mapped to the effective permissions of the connecting user, as evaluated by smbd(8) by reading the unix permissions and POSIX ACL (if present). If the connecting user does not have permission to modify the file, the read only attribute is reported as being set on the file.

? No - The read only DOS attribute is unaffected by permissions, and can only be set by the store dos attributes method. This may be useful for exporting mounted CDs.

Note that this parameter will be ignored if the store dos attributes parameter is set, as the DOS 'read-only' attribute will then be stored inside a UNIX extended attribute.

The default has changed to no in Samba release 4.9.0 and above to allow better Windows fileserver compatibility in a default install.

In addition the default setting of store dos attributes has been changed to Yes in Samba release 4.9.0 and above.

Default: map readonly = no

map system (S)

This controls whether DOS style system files should be mapped to the UNIX group execute bit.

Note that this parameter will be ignored if the store dos attributes parameter is set, as the DOS system attribute will then be stored inside a UNIX extended attribute.

Note that this requires the create mask to be set such that the group execute bit is not masked out (i.e. it must include 010). See the parameter create mask for details.

Default: map system = no

map to guest (G)

This parameter can take four different values, which tell smbd(8) what to do with user login requests that don't match a valid UNIX user in some way.

The four settings are :

? Never - Means user login requests with an invalid password are rejected. This is the default.

? Bad User - Means user logins with an invalid password are rejected, unless the username does not exist, in which case it is treated as a guest login and mapped into the guest account.

? Bad Password - Means user logins with an invalid password are treated as a guest login and mapped into the guest account. Note that this can cause problems as it means that any user incorrectly typing their password will be silently logged on as "guest" - and will not know the reason they cannot access files they think they should - there will have been no message given to them that they got their password wrong. Helpdesk services will hate you if you set the map to guest parameter this way :-).

? Bad Uid - Is only applicable when Samba is configured in some type of domain mode security (security = {domain|ads}) and means that user logins which are successfully authenticated but which have no valid Unix user account (and smbd is unable to create one) should be mapped to the defined guest account. This was the default behavior of Samba 2.x releases. Note that if a member server is running winbindd, this option should never be required because the nss_winbind library will export the Windows domain users and groups to the underlying OS via the Name Service Switch interface.

Note that this parameter is needed to set up "Guest" share services. This is because in these modes the name of the resource being requested is not sent to the server until after the server has successfully authenticated the client so the server cannot make authentication decisions at the correct time (connection to the share) for "Guest" shares.

Default: map to guest = Never

Example: map to guest = Bad User

max connections (S)

This option allows the number of simultaneous connections to a service to be limited. If max connections is greater than 0 then connections will be refused if this number of connections to the service are already open. A value of zero mean an unlimited number of connections may be made.

Record lock files are used to implement this feature. The lock files will be stored in the directory specified by the lock directory option.

Default: max connections = 0

Example: max connections = 10

max disk size (G)

This option allows you to put an upper limit on the apparent size of disks. If you set this option to 100 then all shares will appear to be not larger than 100 MB in size.

Note that this option does not limit the amount of data you can put on the disk. In the above case you could still store much more than 100 MB on the disk, but if a client ever asks for the amount of free disk space or the total disk size then the result will be bounded by the amount specified in max disk size.

This option is primarily useful to work around bugs in some pieces of software that can't handle very large disks, particularly disks over 1GB in size.

A max disk size of 0 means no limit.

Default: max disk size = 0

Example: max disk size = 1000

max log size (G)

This option (an integer in kilobytes) specifies the max size the log file should grow to. Samba periodically checks the size and if it is exceeded it will rename the file, adding a .old extension.

A size of 0 means no limit.

Default: max log size = 5000

Example: max log size = 1000

max mux (G)

This option controls the maximum number of outstanding simultaneous SMB operations that Samba tells the client it will allow. You should never need to set this parameter.

Default: max mux = 50

max open files (G)

This parameter limits the maximum number of open files that one smbd(8) file serving process may have open for a client at any one time. This parameter can be set very high (16384) as Samba uses only one bit per unopened file. Setting this parameter lower than 16384 will cause Samba to complain and set this value back to the minimum of 16384, as Windows 7 depends on this number of open file handles being available.

The limit of the number of open files is usually set by the UNIX per-process file descriptor limit rather than this parameter so you should never need to touch this parameter.

Default: max open files = 16384

max print jobs (S)

This parameter limits the maximum number of jobs allowable in a Samba printer queue at any given moment. If this number is exceeded, smbd(8) will remote "Out of Space" to the client.

Default: max print jobs = 1000

Example: max print jobs = 5000

max reported print jobs (S)

This parameter limits the maximum number of jobs displayed in a port monitor for Samba printer queue at any given moment. If this number is exceeded, the excess jobs will not be shown. A value of zero means there is no limit on the number of print jobs reported.

Default: max reported print jobs = 0

Example: max reported print jobs = 1000

max smbd processes (G)

This parameter limits the maximum number of smbd(8) processes concurrently running on a system and is intended as a stopgap to prevent degrading service to clients in the event that the server has insufficient resources to handle more than this number of connections. Remember that under normal operating conditions, each user will have an smbd(8) associated with him or her to handle connections to all shares from a given host.

For a Samba ADDC running the standard process model this option limits the number of processes forked to handle requests. Currently new processes are only forked for ldap and netlogon requests.

Default: max smbd processes = 0

Example: max smbd processes = 1000

max stat cache size (G)

This parameter limits the size in memory of any stat cache being used to speed up case insensitive name mappings. It represents the number of kilobyte (1024) units the stat cache can use. A value of zero, meaning unlimited, is not advisable due to increased memory usage. You should not need to change this parameter.

Default: max stat cache size = 512

Example: max stat cache size = 100

max ttl (G)

This option tells nmbd(8) what the default 'time to live' of NetBIOS names should be (in seconds) when nmbd is requesting a name using either a broadcast packet or from a WINS server. You should never need to change this parameter. The default is 3 days.

Default: max ttl = 259200

max wins ttl (G)

This option tells smbd(8) when acting as a WINS server (wins support = yes) what the maximum 'time to live' of NetBIOS names that nmbd will grant will be (in seconds). You should never need to change this parameter. The default is 6 days (518400 seconds).

Default: max wins ttl = 518400

max xmit (G)

This option controls the maximum packet size that will be negotiated by Samba's smbd(8) for the SMB1 protocol. The default is 16644, which matches the behavior of Windows 2000. A value below 2048 is likely to cause problems. You should never need to change this parameter from its default value.

Default: max xmit = 16644

Example: max xmit = 8192

mdns name (G)

This parameter controls the name that multicast DNS support advertises as its' hostname.

The default is to use the NETBIOS name which is typically the hostname in all capital letters.

A setting of mdns will defer the hostname configuration to the MDNS

library that is used.

Default: mdns name = netbios

message command (G)

This specifies what command to run when the server receives a

WinPopup style message.

This would normally be a command that would deliver the message

somehow. How this is to be done is up to your imagination.

An example is:

```
message command = csh -c 'xedit %s;rm %s' &
```

This delivers the message using xedit, then removes it afterwards.

NOTE THAT IT IS VERY IMPORTANT THAT THIS COMMAND RETURN

IMMEDIATELY. That's why I have the '&' on the end. If it doesn't

return immediately then your PCs may freeze when sending messages

(they should recover after 30 seconds, hopefully).

All messages are delivered as the global guest user. The command

takes the standard substitutions, although

%u won't work (%U may be better in this case).

Apart from the standard substitutions, some additional ones apply.

In particular:

- ? %s = the filename containing the message.
- ? %t = the destination that the message was sent to

  (probably the server name).
- ? %f = who the message is from.

You could make this command send mail, or whatever else takes your

fancy. Please let us know of any really interesting ideas you have.

Here's a way of sending the messages as mail to root:

```
message command = /bin/mail -s 'message from %f on %m' root < %s; rm %s
```

If you don't have a message command then the message won't be

delivered and Samba will tell the sender there was an error.

Unfortunately WfWg totally ignores the error code and carries on

regardless, saying that the message was delivered.

If you want to silently delete it then try:

```
message command = rm %s
```

Default: message command =

Example: message command = csh -c 'xedit %s; rm %s' &

min domain uid (G)

The integer parameter specifies the minimum uid allowed when mapping a local account to a domain account.

Note that this option interacts with the configured idmap ranges!

Default: min domain uid = 1000

min print space (S)

This sets the minimum amount of free disk space that must be available before a user will be able to spool a print job. It is specified in kilobytes. The default is 0, which means a user can always spool a print job.

Default: min print space = 0

Example: min print space = 2000

min receivefile size (G)

This option changes the behavior of smbd(8) when processing SMBwriteX calls. Any incoming SMBwriteX call on a non-signed SMB/CIFS connection greater than this value will not be processed in the normal way but will be passed to any underlying kernel recvfile or splice system call (if there is no such call Samba will emulate in user space). This allows zero-copy writes directly from network socket buffers into the filesystem buffer cache, if available. It may improve performance but user testing is recommended. If set to zero Samba processes SMBwriteX calls in the normal way. To enable POSIX large write support (SMB/CIFS writes up to 16Mb) this option must be nonzero. The maximum value is 128k. Values greater than 128k will be silently set to 128k.

Note this option will have NO EFFECT if set on a SMB signed connection.

The default is zero, which disables this option.

Default: min receivefile size = 0

min wins ttl (G)

This option tells nmbd(8) when acting as a WINS server (wins support = yes) what the minimum 'time to live' of NetBIOS names that nmbd will grant will be (in seconds). You should never need to change this parameter. The default is 6 hours (21600 seconds).

Default: min wins ttl = 21600

mit kdc command (G)

This option specifies the path to the MIT kdc binary.

If the KDC is not installed in the default location and wasn't correctly detected during build then you should modify this variable and point it to the correct binary.

Default: mit kdc command =

Example: mit kdc command = /opt/mit/sbin/krb5kdc

msdfs proxy (S)

This parameter indicates that the share is a stand-in for another CIFS share whose location is specified by the value of the parameter. When clients attempt to connect to this share, they are redirected to one or multiple, comma separated proxied shares using the SMB-Dfs protocol.

Only Dfs roots can act as proxy shares. Take a look at the msdfs root and host msdfs options to find out how to set up a Dfs root share.

No default

Example: msdfs proxy = \otherserver\someshare,\otherserver2\someshare

msdfs root (S)

If set to yes, Samba treats the share as a Dfs root and allows clients to browse the distributed file system tree rooted at the share directory. Dfs links are specified in the share directory by symbolic links of the form msdfs:serverA\\shareA,serverB\\shareB and so on. For more information on setting up a Dfs tree on Samba, refer to the MSDFS chapter in the Samba3-HOWTO book.

Default: msdfs root = no

msdfs shuffle referrals (S)

If set to yes, Samba will shuffle Dfs referrals for a given Dfs link if multiple are available, allowing for load balancing across clients. For more information on setting up a Dfs tree on Samba, refer to the MSDFS chapter in the Samba3-HOWTO book.

Default: msdfs shuffle referrals = no

multicast dns register (G)

If compiled with proper support for it, Samba will announce itself with multicast DNS services like for example provided by the Avahi daemon.

This parameter allows disabling Samba to register itself.

Default: multicast dns register = yes

name cache timeout (G)

Specifies the number of seconds it takes before entries in samba's hostname resolve cache time out. If the timeout is set to 0. the caching is disabled.

Default: name cache timeout = 660

Example: name cache timeout = 0

name resolve order (G)

This option is used by the programs in the Samba suite to determine what naming services to use and in what order to resolve host names to IP addresses. Its main purpose to is to control how netbios name resolution is performed. The option takes a space separated string of name resolution options.

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

? 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 manpage for lmhosts for details) then any name type matches for lookup.

? host : Do a standard host name to IP address resolution, using the system /etc/hosts or DNS lookups. This method of name resolution is operating system depended for instance on IRIX or Solaris this may be controlled by the /etc/nsswitch.conf file. Note that this method is used only if the NetBIOS name type being queried is the 0x20 (server) name type or 0x1c (domain controllers). The latter case is only useful for active directory domains and results in a DNS query for the SRV RR entry matching _ldap._tcp.domain.

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

? 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.

The example below will cause the local lmhosts file to be examined first, followed by a broadcast attempt, followed by a normal system hostname lookup.

When Samba is functioning in ADS security mode (security = ads) it is advised to use following settings for name resolve order:

name resolve order = wins bcast

DC lookups will still be done via DNS, but fallbacks to netbios names will not inundate your DNS servers with needless queries for DOMAIN<0x1c> lookups.

Default: name resolve order = lmhosts wins host bcast

Example: name resolve order = lmhosts bcast host

socket address

This parameter is a synonym for nbt client socket address.

nbt client socket address (G)

This option allows you to control what address Samba will send NBT client packets from, and process replies using, including in nmbd.

Setting this option should never be necessary on usual Samba servers running only one nmbd.

By default Samba will send UDP packets from the OS default address for the destination, and accept replies on 0.0.0.0.

This parameter is deprecated. See bind interfaces only = Yes and interfaces for the previous behaviour of controlling the normal listening sockets.

Default: nbt client socket address = 0.0.0.0

Example: nbt client socket address = 192.168.2.20

nbtd:wins_prepend1Bto1Cqueries (G)

Normally queries for 0x1C names (all logon servers for a domain) will return the first address of the 0x1B names (domain master browser and PDC) as first address in the result list. As many client only use the first address in the list by default, all clients will use the same server (the PDC). Windows servers have an option to disable this behavior (since Windows 2000 Service Pack 2).

Default: nbtd:wins_prepend1Bto1Cqueries = yes

nbtd:wins_wins_randomize1Clist (G)

Normally queries for 0x1C names will return the addresses in the same order as they're stored in the database, that means first all addresses which have been directly registered at the local wins server and then all addresses registered at other servers. Windows servers have an option to change this behavior and randomize the returned addresses. Set this parameter to "yes" and Samba will sort the address list depending on the client address and the matching bits of the addresses, the first address is randomized based on depending on the "nbtd:wins_randomize1Clist_mask" parameter.

Default: nbtd:wins_wins_randomize1Clist = no

nbtd:wins_randomize1Clist_mask (G)

If the "nbtd:wins_randomize1Clist" parameter is set to "yes", then randomizing of the first returned address is based on the specified netmask. If there are addresses which are in the same subnet as the client address, the first returned address is randomly chosen out them. Otherwise the first returned address is randomly chosen out of all addresses.

Default: nbtd:wins_randomize1Clist_mask = 255.255.255.0

nbt port (G)

Specifies which port the server should use for NetBIOS over IP name services traffic.

Default: nbt port = 137

ncalrpc dir (G)

This directory will hold a series of named pipes to allow RPC over inter-process communication.

This will allow Samba and other unix processes to interact over DCE/RPC without using TCP/IP. Additionally a sub-directory 'np' has restricted permissions, and allows a trusted communication channel between Samba processes

Default: ncalrpc dir = /run/samba/ncalrpc

Example: ncalrpc dir = /var/run/samba/ncalrpc

netbios aliases (G)

This is a list of NetBIOS names that nmbd will advertise as additional names by which the Samba server is known. This allows one machine to appear in browse lists under multiple names. If a machine is acting as a browse server or logon server none of these names will be advertised as either browse server or logon servers, only the primary name of the machine will be advertised with these capabilities.

Default: netbios aliases = # empty string (no additional names)

Example: netbios aliases = TEST TEST1 TEST2

netbios name (G)

This sets the NetBIOS name by which a Samba server is known. By default it is the same as the first component of the host's DNS name. If a machine is a browse server or logon server this name (or the first component of the hosts DNS name) will be the name that these services are advertised under.

Note that the maximum length for a NetBIOS name is 15 characters.

There is a bug in Samba that breaks operation of browsing and access to shares if the netbios name is set to the literal name

PIPE. To avoid this problem, do not name your Samba server PIPE.

Default: netbios name = # machine DNS name

Example: netbios name = MYNAME

netbios scope (G)

This sets the NetBIOS scope that Samba will operate under. This should not be set unless every machine on your LAN also sets this value.

Default: netbios scope =

neutralize nt4 emulation (G)

This option controls whether winbindd sends the NETLOGON_NEG_NEUTRALIZE_NT4_EMULATION flag in order to bypass the NT4 emulation of a domain controller.

Typically you should not need set this. It can be useful for upgrades from NT4 to AD domains.

The behavior can be controlled per netbios domain by using 'neutralize nt4 emulation:NETBIOSDOMAIN = yes' as option.

Default: neutralize nt4 emulation = no

nmbd bind explicit broadcast (G)

This option causes nmbd(8) to explicitly bind to the broadcast address of the local subnets. This is needed to make nmbd work correctly in combination with the socket address option. You should not need to unset this option.

Default: nmbd bind explicit broadcast = yes

nsupdate command (G)

This option sets the path to the nsupdate command which is used for GSS-TSIG dynamic DNS updates.

Default: nsupdate command = /usr/bin/nsupdate -g

nt hash store (G)

This parameter determines whether or not samba(8) will, as an AD DC, attempt to store the NT password hash used in NTLM and NTLMv2 authentication for users in this domain.

If so configured, the Samba Active Directory Domain Controller, will, except for trust accounts (computers, domain controllers and inter-domain trusts) the NOT store the NT hash for new and changed accounts in the sam.ldb database.

This avoids the storage of an unsalted hash for these user-created passwords. As a consequence the arcfour-hmac-md5 Kerberos key type is also unavailable in the KDC for these users - thankfully modern clients will select an AES based key instead.

NOTE: As the password history in Active Directory is stored as an NT hash (and thus unavailable), a workaround is used, relying instead on Kerberos password hash values. This stores three passwords, the current, previous and second previous password. This allows some checking against reuse.

However as these values are salted, changing the sAMAccountName, userAccountControl or userPrincipalName of an account will cause the salt to change. After the rare combination of both a rename and a password change only the current password will be recognised for password history purposes.

The available settings are:

? always - Always store the NT hash (as machine accounts will also always store an NT hash, a hash will be stored for all accounts).

This setting may be useful if ntlm auth is set to disabled for a trial period

? never - Never store the NT hash for user accounts, only for machine accounts

? auto - Store an NT hash if ntlm auth is not set to disabled.

Default: nt hash store = always

nt acl support (S)

This boolean parameter controls whether smbd(8) will attempt to map UNIX permissions into Windows NT access control lists. The UNIX permissions considered are the traditional UNIX owner and group permissions, as well as POSIX ACLs set on any files or directories.

This parameter was formally a global parameter in releases prior to 2.2.2.

Default: nt acl support = yes

ntlm auth (G)

This parameter determines whether or not smbd(8) will attempt to authenticate users using the NTLM encrypted password response for this local passdb (SAM or account database).

If disabled, both NTLM and LanMan authentication against the local passdb is disabled.

Note that these settings apply only to local users, authentication will still be forwarded to and NTLM authentication accepted against any domain we are joined to, and any trusted domain, even if disabled or if NTLMv2-only is enforced here. To control NTLM authentiation for domain users, this must option must be configured on each DC.

By default with ntlm auth set to ntlmv2-only only NTLMv2 logins will be permitted. All modern clients support NTLMv2 by default, but some older clients will require special configuration to use it.

The primary user of NTLMv1 is MSCHAPv2 for VPNs and 802.1x.

The available settings are:

? ntlmv1-permitted (alias yes) - Allow NTLMv1 and above for all clients.

This is the required setting for to enable the lanman auth parameter.

? ntlmv2-only (alias no) - Do not allow NTLMv1 to be used, but permit NTLMv2.

? mschapv2-and-ntlmv2-only - Only allow NTLMv1 when the client promises that it is providing MSCHAPv2 authentication (such as the ntlm_auth tool).

? disabled - Do not accept NTLM (or LanMan) authentication of any level, nor permit NTLM password changes.

WARNING: Both Microsoft Windows and Samba Read Only

Domain Controllers (RODCs) convert a plain-text LDAP Simple Bind into an NTLMv2 authentication to forward to a full DC. Setting this option to disabled will cause these forwarded authentications to fail.

Additionally, for Samba acting as an Active Directory Domain Controller, for user accounts, if nt hash store is set to the default setting of auto, the NT hash will not be stored in the sam.ldb database for new users and after a password change.

The default changed from yes to no with Samba 4.5. The default changed again to ntlmv2-only with Samba 4.7, however the behaviour is unchanged.

Default: ntlm auth = ntlmv2-only

nt pipe support (G)

This boolean parameter controls whether smbd(8) will allow Windows NT clients to connect to the NT SMB specific IPC$ pipes. This is a developer debugging option and can be left alone.

Default: nt pipe support = yes

ntp signd socket directory (G)

This setting controls the location of the socket that the NTP daemon uses to communicate with Samba for signing packets.

If a non-default path is specified here, then it is also necessary to make NTP aware of the new path using the ntpsigndsocket directive in ntp.conf.

Default: ntp signd socket directory = /var/lib/samba/ntp_signd

nt status support (G)

This boolean parameter controls whether smbd(8) will negotiate NT specific status support with Windows NT/2k/XP clients. This is a developer debugging option and should be left alone. If this option is set to no then Samba offers exactly the same DOS error codes that versions prior to Samba 2.2.3 reported.

You should not need to ever disable this parameter.

Default: nt status support = yes

ntvfs handler (S)

This specifies the NTVFS handlers for this share.

? unixuid: Sets up user credentials based on POSIX gid/uid.

? cifs: Proxies a remote CIFS FS. Mainly useful for testing.

? nbench: Filter module that saves data useful to the nbench benchmark suite.

? ipc: Allows using SMB for inter process communication. Only used for the IPC$ share.

? posix: Maps POSIX FS semantics to NT semantics

? print: Allows printing over SMB. This is LANMAN-style printing, not the be confused with the spoolss DCE/RPC interface used by later versions of Windows.

Note that this option is only used when the NTVFS file server is in use. It is not used with the (default) s3fs file server.

Default: ntvfs handler = unixuid, default

null passwords (G)

Allow or disallow client access to accounts that have null passwords.

See also smbpasswd(5).

Default: null passwords = no

obey pam restrictions (G)

When Samba 3.0 is configured to enable PAM support (i.e. --with-pam), this parameter will control whether or not Samba should obey PAM's account and session management directives. The default behavior is to use PAM for clear text authentication only and to ignore any account or session management. Note that Samba always ignores PAM for authentication in the case of encrypt passwords = yes. The reason is that PAM modules cannot support the challenge/response authentication mechanism needed in the presence of SMB password encryption.

Default: obey pam restrictions = no

old password allowed period (G)

Number of minutes to permit an NTLM login after a password change or reset using the old password. This allows the user to re-cache the new password on multiple clients without disrupting a network reconnection in the meantime.

This parameter only applies when server role is set to Active Directory Domain Controller.

Default: old password allowed period = 60

oplock break wait time (G)

This is a tuning parameter added due to bugs in both Windows 9x and WinNT. If Samba responds to a client too quickly when that client issues an SMB that can cause an oplock break request, then the network client can fail and not respond to the break request. This tuning parameter (which is set in milliseconds) is the amount of time Samba will wait before sending an oplock break request to such (broken) clients.

Warning

DO NOT CHANGE THIS PARAMETER UNLESS YOU HAVE READ AND UNDERSTOOD THE SAMBA OPLOCK CODE.

Default: oplock break wait time = 0

oplocks (S)

This boolean option tells smbd whether to issue oplocks (opportunistic locks) to file open requests on this share. The oplock code can dramatically (approx. 30% or more) improve the speed of access to files on Samba servers. It allows the clients to aggressively cache files locally and you may want to disable this option for unreliable network environments (it is turned on by default in Windows NT Servers).

Oplocks may be selectively turned off on certain files with a share. See the veto oplock files parameter. On some systems oplocks are recognized by the underlying operating system. This allows data synchronization between all access to oplocked files, whether it be via Samba or NFS or a local UNIX process. See the kernel oplocks parameter for details.

Default: oplocks = yes

os2 driver map (G)

The parameter is used to define the absolute path to a file containing a mapping of Windows NT printer driver names to OS/2 printer driver names. The format is:

<nt driver name> = <os2 driver name>.<device name>

For example, a valid entry using the HP LaserJet 5 printer driver would appear as HP LaserJet 5L = LASERJET.HP LaserJet 5L.

The need for the file is due to the printer driver namespace problem described in the chapter on Classical Printing in the Samba3-HOWTO book. For more details on OS/2 clients, please refer to chapter on other clients in the Samba3-HOWTO book.

Default: os2 driver map =

os level (G)

This integer value controls what level Samba advertises itself as for browse elections. The value of this parameter determines whether nmbd(8) has a chance of becoming a local master browser for the workgroup in the local broadcast area.

Note: By default, Samba will win a local master browsing election over all Microsoft operating systems except a Windows NT 4.0/2000 Domain Controller. This means that a misconfigured Samba host can effectively isolate a subnet for browsing purposes. This parameter is largely auto-configured in the Samba-3 release series and it is seldom necessary to manually override the default setting. Please refer to the chapter on Network Browsing in the Samba-3 HOWTO document for further information regarding the use of this parameter. Note: The maximum value for this parameter is 255. If you use higher values, counting will start at 0!

Default: os level = 20

Example: os level = 65

pam password change (G)

With the addition of better PAM support in Samba 2.2, this parameter, it is possible to use PAM's password change control flag for Samba. If enabled, then PAM will be used for password changes when requested by an SMB client instead of the program listed in passwd program. It should be possible to enable this without changing your passwd chat parameter for most setups.

Default: pam password change = no

panic action (G)

This is a Samba developer option that allows a system command to be called when either smbd(8) or nmbd(8) crashes. This is usually used to draw attention to the fact that a problem occurred.

Default: panic action =

Example: panic action = /bin/sleep 90000

passdb backend (G)

This option allows the administrator to chose which backend will be used for storing user and possibly group information. This allows you to swap between different storage mechanisms without recompile.

The parameter value is divided into two parts, the backend's name, and a 'location' string that has meaning only to that particular backed. These are separated by a : character.

Available backends can include:

? smbpasswd - The old plaintext passdb backend. Some Samba features will not work if this passdb backend is used. Takes a path to the smbpasswd file as an optional argument.

? tdbsam - The TDB based password storage backend. Takes a path to the TDB as an optional argument (defaults to passdb.tdb in the private dir directory.

? ldapsam - The LDAP based passdb backend. Takes an LDAP URL as an optional argument (defaults to ldap://localhost)

LDAP connections should be secured where possible. This may be done using either Start-TLS (see ldap ssl) or by specifying ldaps:// in the URL argument.

Multiple servers may also be specified in double-quotes.

Whether multiple servers are supported or not and the exact syntax depends on the LDAP library you use.

Examples of use are:

passdb backend = tdbsam:/etc/samba/private/passdb.tdb

or multi server LDAP URL with OpenLDAP library:

passdb backend = ldapsam:"ldap://ldap-1.example.com ldap://ldap-2.example.com"

or multi server LDAP URL with Netscape based LDAP library:

passdb backend = ldapsam:"ldap://ldap-1.example.com ldap-2.example.com"

Default: passdb backend = tdbsam

passdb expand explicit (G)

This parameter controls whether Samba substitutes %-macros in the passdb fields if they are explicitly set. We used to expand macros here, but this turned out to be a bug because the Windows client can expand a variable %G_osver% in which %G would have been substituted by the user's primary group.

Default: passdb expand explicit = no

passwd chat (G)

This string controls the "chat" conversation that takes places between smbd(8) and the local password changing program to change the user's password. The string describes a sequence of response-receive pairs that smbd(8) uses to determine what to send to the passwd program and what to expect back. If the expected output is not received then the password is not changed.

This chat sequence is often quite site specific, depending on what local methods are used for password control.

Note that this parameter only is used if the unix password sync parameter is set to yes. This sequence is then called AS ROOT when the SMB password in the smbpasswd file is being changed, without access to the old password cleartext. This means that root must be able to reset the user's password without knowing the text of the previous password.

The string can contain the macro %n which is substituted for the new password. The old password (%o) is only available when encrypt passwords has been disabled. The chat sequence can also contain the standard macros \n, \r, \t and \s to give line-feed, carriage-return, tab and space. The chat sequence string can also contain a '*' which matches any sequence of characters. Double quotes can be used to collect strings with spaces in them into a single string.

If the send string in any part of the chat sequence is a full stop ".", then no string is sent. Similarly, if the expect string is a full stop then no string is expected.

If the pam password change parameter is set to yes, the chat pairs may be matched in any order, and success is determined by the PAM result, not any particular output. The \n macro is ignored for PAM conversions.

Default: passwd chat = *new*password* %n\n *new*password* %n\n *changed*

Example: passwd chat = "*Enter NEW password*" %n\n "*Reenter NEW password*" %n\n "*Password changed*"

passwd chat debug (G)

This boolean specifies if the passwd chat script parameter is run in debug mode. In this mode the strings passed to and received from the passwd chat are printed in the smbd(8) log with a debug level of 100. This is a dangerous option as it will allow plaintext passwords to be seen in the smbd log. It is available to help Samba admins debug their passwd chat scripts when calling the passwd program and should be turned off after this has been done. This option has no effect if the pam password change parameter is set. This parameter is off by default.

Default: passwd chat debug = no

passwd chat timeout (G)

This integer specifies the number of seconds smbd will wait for an initial answer from a passwd chat script being run. Once the initial answer is received the subsequent answers must be received in one tenth of this time. The default it two seconds.

Default: passwd chat timeout = 2

passwd program (G)

The name of a program that can be used to set UNIX user passwords. Any occurrences of %u will be replaced with the user name. The user name is checked for existence before calling the password changing program.

Also note that many passwd programs insist in reasonable passwords, such as a minimum length, or the inclusion of mixed case chars and digits. This can pose a problem as some clients (such as Windows for Workgroups) uppercase the password before sending it.

Note that if the unix password sync parameter is set to yes then this program is called AS ROOT before the SMB password in the smbpasswd file is changed. If this UNIX password change fails, then smbd will fail to change the SMB password also (this is by design).

If the unix password sync parameter is set this parameter MUST USE ABSOLUTE PATHS for ALL programs called, and must be examined for security implications. Note that by default unix password sync is set to no.

Default: passwd program =

Example: passwd program = /bin/passwd %u

password hash gpg key ids (G)

If samba is running as an active directory domain controller, it is possible to store the cleartext password of accounts in a PGP/OpenGPG encrypted form.

You can specify one or more recipients by key id or user id. Note that 32bit key ids are not allowed, specify at least 64bit.

The value is stored as 'Primary:SambaGPG' in the supplementalCredentials attribute.

As password changes can occur on any domain controller, you should configure this on each of them. Note that this feature is currently available only on Samba domain controllers.

This option is only available if samba was compiled with gpgme support.

You may need to export the GNUPGHOME environment variable before starting samba. It is strongly recommended to only store the public key in this location. The private key is not used for encryption and should be only stored where decryption is required.

Being able to restore the cleartext password helps, when they need to be imported into other authentication systems later (see samba-tool user getpassword) or you want to keep the passwords in sync with another system, e.g. an OpenLDAP server (see samba-tool user syncpasswords).

While this option needs to be configured on all domain controllers, the samba-tool user syncpasswords command should run on a single domain controller only (typically the PDC-emulator).

Default: password hash gpg key ids =

Example: password hash gpg key ids = 4952E40301FAB41A

Example: password hash gpg key ids = selftest@samba.example.com

Example: password hash gpg key ids = selftest@samba.example.com, 4952E40301FAB41A

password hash userPassword schemes (G)

This parameter determines whether or not samba(8) acting as an Active Directory Domain Controller will attempt to store additional passwords hash types for the user

The values are stored as 'Primary:userPassword' in the supplementalCredentials attribute. The value of this option is a hash type.

The currently supported hash types are:

? CryptSHA256

? CryptSHA512

Multiple instances of a hash type may be computed and stored. The password hashes are calculated using the crypt(3) call. The number of rounds used to compute the hash can be specified by adding ':rounds=xxxx' to the hash type, i.e. CryptSHA512:rounds=4500 would calculate an SHA512 hash using 4500 rounds. If not specified the

Operating System defaults for crypt(3) are used.

As password changes can occur on any domain controller, you should configure this on each of them. Note that this feature is currently available only on Samba domain controllers.

Currently the NT Hash of the password is recorded when these hashes are calculated and stored. When retrieving the hashes the current value of the NT Hash is checked against the stored NT Hash. This detects password changes that have not updated the password hashes. In this case samba-tool user will ignore the stored hash values.

Being able to obtain the hashed password helps, when they need to be imported into other authentication systems later (see samba-tool user getpassword) or you want to keep the passwords in sync with another system, e.g. an OpenLDAP server (see samba-tool user syncpasswords).

Related command: unix password sync

Default: password hash userPassword schemes =

Example: password hash userPassword schemes = CryptSHA256

Example: password hash userPassword schemes = CryptSHA256 CryptSHA512

Example: password hash userPassword schemes = CryptSHA256:rounds=5000 CryptSHA512:rounds=7000

password server (G)

By specifying the name of a domain controller with this option, and using security = [ads|domain] it is possible to get Samba to do all its username/password validation using a specific remote server.

Ideally, this option should not be used, as the default '*' indicates to Samba to determine the best DC to contact dynamically, just as all other hosts in an AD domain do. This allows the domain to be maintained (addition and removal of domain controllers) without modification to the smb.conf file. The cryptographic protection on the authenticated RPC calls used to verify passwords ensures that this default is safe.

It is strongly recommended that you use the default of '*', however if in your particular environment you have reason to specify a particular DC list, then the list of machines in this option must be a list of names or IP addresses of Domain controllers for the Domain. If you use the default of '*', or list several hosts in the password server option then smbd will try each in turn till it finds one that responds. This is useful in case your primary server goes down.

If the list of servers contains both names/IP's and the '*' character, the list is treated as a list of preferred domain controllers, but an auto lookup of all remaining DC's will be added to the list as well. Samba will not attempt to optimize this list by locating the closest DC.

If parameter is a name, it is looked up using the parameter name resolve order and so may resolved by any method and order described in that parameter.

Default: password server = *

Example: password server = NT-PDC, NT-BDC1, NT-BDC2, *

Example: password server = windc.mydomain.com:389 192.168.1.101 *

directory

This parameter is a synonym for path.

path (S)

This parameter specifies a directory to which the user of the service is to be given access. In the case of printable services, this is where print data will spool prior to being submitted to the host for printing.

For a printable service offering guest access, the service should be readonly and the path should be world-writeable and have the sticky bit set. This is not mandatory of course, but you probably won't get the results you expect if you do otherwise.

Any occurrences of %u in the path will be replaced with the UNIX username that the client is using on this connection. Any occurrences of %m will be replaced by the NetBIOS name of the machine they are connecting from. These replacements are very useful for setting up pseudo home directories for users.

Note that this path will be based on root dir if one was specified.

Default: path =

Example: path = /home/fred

perfcount module (G)

This parameter specifies the perfcount backend to be used when monitoring SMB operations. Only one perfcount module may be used, and it must implement all of the apis contained in the smb_perfcount_handler structure defined in smb.h.

No default

pid directory (G)

This option specifies the directory where pid files will be placed.

Default: pid directory = /run

Example: pid directory = /var/run/

posix locking (S)

The smbd(8) daemon maintains an database of file locks obtained by SMB clients. The default behavior is to map this internal database to POSIX locks. This means that file locks obtained by SMB clients are consistent with those seen by POSIX compliant applications accessing the files via a non-SMB method (e.g. NFS or local file access). It is very unlikely that you need to set this parameter to "no", unless you are sharing from an NFS mount, which is not a good idea in the first place.

Default: posix locking = yes

postexec (S)

This option specifies a command to be run whenever the service is disconnected. It takes the usual substitutions. The command may be run as the root on some systems.

An interesting example may be to unmount server resources:

postexec = /etc/umount /cdrom

Default: postexec =

Example: postexec = echo \"%u disconnected from %S from %m (%I)\" >> /tmp/log exec

This parameter is a synonym for preexec.

preexec (S)

This option specifies a command to be run whenever the service is connected to. It takes the usual substitutions.

An interesting example is to send the users a welcome message every time they log in. Maybe a message of the day? Here is an example:

```
preexec = csh -c 'echo \"Welcome to %S!\" |
/usr/local/samba/bin/smbclient -M %m -I %I' &
```

Of course, this could get annoying after a while :-)

See also preexec close and postexec.

Default: preexec =

Example: preexec = echo \"%u connected to %S from %m (%I)\" >> /tmp/log

preexec close (S)

This boolean option controls whether a non-zero return code from preexec should close the service being connected to.

Default: preexec close = no

prefered master

This parameter is a synonym for preferred master.

preferred master (G)

This boolean parameter controls if nmbd(8) is a preferred master browser for its workgroup.

If this is set to yes, on startup, nmbd will force an election, and it will have a slight advantage in winning the election. It is recommended that this parameter is used in conjunction with domain master = yes, so that nmbd can guarantee becoming a domain master.

Use this option with caution, because if there are several hosts (whether Samba servers, Windows 95 or NT) that are preferred master browsers on the same subnet, they will each periodically and continuously attempt to become the local master browser. This will result in unnecessary broadcast traffic and reduced browsing capabilities.

Default: preferred master = auto

prefork backoff increment (G)

This option specifies the number of seconds added to the delay before a prefork master or worker process is restarted. The restart is initially zero, the prefork backoff increment is added to the delay on each restart up to the value specified by "prefork maximum backoff".

Additionally the the backoff for an individual service by using "prefork backoff increment: service name" i.e. "prefork backoff increment:ldap = 2" to set the backoff increment to 2.

If the backoff increment is 2 and the maximum backoff is 5. There will be a zero second delay for the first restart. A two second delay for the second restart. A four second delay for the third and any subsequent restarts

Default: prefork backoff increment = 10

prefork children (G)

This option controls the number of worker processes that are started for each service when prefork process model is enabled (see samba(8) -M) The prefork children are only started for those services that support prefork (currently ldap, kdc and netlogon).

For processes that don't support preforking all requests are handled by a single process for that service.

This should be set to a small multiple of the number of CPU's available on the server

Additionally the number of prefork children can be specified for an individual service by using "prefork children: service name" i.e. "prefork children:ldap = 8" to set the number of ldap worker processes.

Default: prefork children = 4

prefork maximum backoff (G)

This option controls the maximum delay before a failed pre-fork process is restarted.

Default: prefork maximum backoff = 120

preload modules (G)

This is a list of paths to modules that should be loaded into smbd before a client connects. This improves the speed of smbd when reacting to new connections somewhat.

Default: preload modules =

Example: preload modules = /usr/lib/samba/passdb/mysql.so

preserve case (S)

This controls if new filenames are created with the case that the client passes, or if they are forced to be the default case.

See the section on NAME MANGLING for a fuller discussion.

Default: preserve case = yes

print ok

This parameter is a synonym for printable.

printable (S)

If this parameter is yes, then clients may open, write to and submit spool files on the directory specified for the service.

Note that a printable service will ALWAYS allow writing to the service path (user privileges permitting) via the spooling of print data. The read only parameter controls only non-printing access to the resource.

Default: printable = no

printcap cache time (G)

This option specifies the number of seconds before the printing subsystem is again asked for the known printers.

Setting this parameter to 0 disables any rescanning for new or removed printers after the initial startup.

Default: printcap cache time = 750

Example: printcap cache time = 600

printcap

This parameter is a synonym for printcap name.

printcap name (G)

This parameter may be used to override the compiled-in default printcap name used by the server (usually /etc/printcap). See the discussion of the [printers] section above for reasons why you might want to do this.

To use the CUPS printing interface set printcap name = cups. This should be supplemented by an additional setting printing = cups in the [global] section. printcap name = cups will use the "dummy" printcap created by CUPS, as specified in your CUPS configuration file.

On System V systems that use lpstat to list available printers you can use printcap name = lpstat to automatically obtain lists of available printers. This is the default for systems that define SYSV at configure time in Samba (this includes most System V based systems). If

printcap name is set to lpstat on these systems then Samba will launch lpstat -v and attempt to parse the output to obtain a printer list.

A minimal printcap file would look something like this:

```
print1|My Printer 1
print2|My Printer 2
print3|My Printer 3
print4|My Printer 4
print5|My Printer 5
```

where the '|' separates aliases of a printer. The fact that the second alias has a space in it gives a hint to Samba that it's a comment.

Note

Under AIX the default printcap name is /etc/qconfig. Samba will assume the file is in AIX qconfig format if the string qconfig appears in the printcap filename.

Default: printcap name = /etc/printcap

Example: printcap name = /etc/myprintcap

print command (S)

After a print job has finished spooling to a service, this command will be used via a system() call to process the spool file.

Typically the command specified will submit the spool file to the host's printing subsystem, but there is no requirement that this be the case. The server will not remove the spool file, so whatever command you specify should remove the spool file when it has been processed, otherwise you will need to manually remove old spool files.

The print command is simply a text string. It will be used verbatim after macro substitutions have been made:

%s, %f - the path to the spool file name

%p - the appropriate printer name

%J - the job name as transmitted by the client.

%c - The number of printed pages of the spooled job (if known).

%z - the size of the spooled print job (in bytes)

The print command MUST contain at least one occurrence of %s or %f - the %p is optional. At the time a job is submitted, if no printer name is supplied the %p will be silently removed from the printer command.

If specified in the [global] section, the print command given will be used for any printable service that does not have its own print command specified.

If there is neither a specified print command for a printable service nor a global print command, spool files will be created but not processed and (most importantly) not removed.

Note that printing may fail on some UNIXes from the nobody account. If this happens then create an alternative guest account that can print and set the guest account in the [global] section.

You can form quite complex print commands by realizing that they are just passed to a shell. For example the following will log a print job, print the file, then remove it. Note that ';' is the usual separator for command in shell scripts.

```
print command = echo Printing %s >> /tmp/print.log; lpr -P %p %s;
rm %s
```

You may have to vary this command considerably depending on how you normally print files on your system. The default for the parameter

varies depending on the setting of the printing parameter.

Default: For printing = BSD, AIX, QNX, LPRNG or PLP :

print command = lpr -r -P%p %s

For printing = SYSV or HPUX :

print command = lp -c -d%p %s; rm %s

For printing = SOFTQ :

print command = lp -d%p -s %s; rm %s

For printing = CUPS : If SAMBA is compiled against libcups, then

printcap = cups uses the CUPS API to submit jobs, etc. Otherwise it

maps to the System V commands with the -oraw option for printing,

i.e. it uses lp -c -d%p -oraw; rm %s. With printing = cups, and if

SAMBA is compiled against libcups, any manually set print command

will be ignored.

No default

Example: print command = /usr/local/samba/bin/myprintscript %p %s

printer

This parameter is a synonym for printer name.

printer name (S)

This parameter specifies the name of the printer to which print

jobs spooled through a printable service will be sent.

If specified in the [global] section, the printer name given will

be used for any printable service that does not have its own

printer name specified.

The default value of the printer name may be lp on many systems.

Default: printer name =

Example: printer name = laserwriter

printing (S)

This parameters controls how printer status information is

interpreted on your system. It also affects the default values for

the print command, lpq command, lppause command , lpresume command,

and lprm command if specified in the [global] section.

Currently nine printing styles are supported. They are BSD, AIX, LPRNG, PLP, SYSV, HPUX, QNX, SOFTQ, CUPS and IPRINT.

Be aware that CUPS and IPRINT are only available if the CUPS development library was available at the time Samba was compiled or packaged.

To see what the defaults are for the other print commands when using the various options use the testparm(1) program.

This option can be set on a per printer basis. Please be aware however, that you must place any of the various printing commands (e.g. print command, lpq command, etc...) after defining the value for the printing option since it will reset the printing commands to default values.

See also the discussion in the [printers] section.

See testparm -v. for the default value on your system

Default: printing = # Depends on the operating system

printjob username (S)

This parameter specifies which user information will be passed to the printing system. Usually, the username is sent, but in some cases, e.g. the domain prefix is useful, too.

Default: printjob username = %U

Example: printjob username = %D\%U

print notify backchannel (S)

Windows print clients can update print queue status by expecting the server to open a backchannel SMB connection to them. Due to client firewall settings this can cause considerable timeouts and will often fail, as there is no guarantee the client is even running an SMB server. By default, the Samba print server will not try to connect back to clients, and will treat corresponding requests as if the connection back to the client failed.

Default: print notify backchannel = no

private directory

This parameter is a synonym for private dir.

private dir (G)

This parameters defines the directory smbd will use for storing such files as smbpasswd and secrets.tdb.

Default: private dir = /var/lib/samba/private

queuepause command (S)

This parameter specifies the command to be executed on the server host in order to pause the printer queue.

This command should be a program or script which takes a printer name as its only parameter and stops the printer queue, such that no longer jobs are submitted to the printer.

This command is not supported by Windows for Workgroups, but can be issued from the Printers window under Windows 95 and NT.

If a %p is given then the printer name is put in its place.

Otherwise it is placed at the end of the command.

Note that it is good practice to include the absolute path in the command as the PATH may not be available to the server.

Default: queuepause command = # determined by printing parameter

Example: queuepause command = disable %p

queueresume command (S)

This parameter specifies the command to be executed on the server host in order to resume the printer queue. It is the command to undo the behavior that is caused by the previous parameter (queuepause command).

This command should be a program or script which takes a printer name as its only parameter and resumes the printer queue, such that queued jobs are resubmitted to the printer.

This command is not supported by Windows for Workgroups, but can be issued from the Printers window under Windows 95 and NT.

If a %p is given then the printer name is put in its place.

Otherwise it is placed at the end of the command.

Note that it is good practice to include the absolute path in the command as the PATH may not be available to the server.

Default: queueresume command = # determined by printing parameter

Example: queueresume command = enable %p

raw NTLMv2 auth (G)

This parameter has been deprecated since Samba 4.13 and support for NTLMv2 authentication without NTLMSSP will be removed in a future Samba release.

That is, in the future, the current default of raw NTLMv2 auth = no will be the enforced behaviour.

This parameter determines whether or not smbd(8) will allow SMB1 clients without extended security (without SPNEGO) to use NTLMv2 authentication.

If this option, lanman auth and ntlm auth are all disabled, then only clients with SPNEGO support will be permitted. That means NTLMv2 is only supported within NTLMSSP.

Default: raw NTLMv2 auth = no

read list (S)

This is a list of users that are given read-only access to a service. If the connecting user is in this list then they will not be given write access, no matter what the read only option is set to. The list can include group names using the syntax described in the invalid users parameter.

Default: read list =

Example: read list = mary, @students

read only (S)

An inverted synonym is writeable.

If this parameter is yes, then users of a service may not create or modify files in the service's directory.

Note that a printable service (printable = yes) will ALWAYS allow writing to the directory (user privileges permitting), but only via spooling operations.

Default: read only = yes

read raw (G)

This is ignored if async smb echo handler is set, because this feature is incompatible with raw read SMB requests

If enabled, raw reads allow reads of 65535 bytes in one packet.

This typically provides a major performance benefit for some very,

very old clients.

However, some clients either negotiate the allowable block size incorrectly or are incapable of supporting larger block sizes, and for these clients you may need to disable raw reads.

In general this parameter should be viewed as a system tuning tool and left severely alone.

Default: read raw = yes

realm (G)

This option specifies the kerberos realm to use. The realm is used as the ADS equivalent of the NT4 domain. It is usually set to the DNS name of the kerberos server.

Default: realm =

Example: realm = mysambabox.mycompany.com

registry shares (G)

This turns on or off support for share definitions read from registry. Shares defined in smb.conf take precedence over shares with the same name defined in registry. See the section on registry-based configuration for details.

Note that this parameter defaults to no, but it is set to yes when config backend is set to registry.

Default: registry shares = no

Example: registry shares = yes

reject md5 clients (G)

This option is deprecated and will be removed in a future release, as it is a security problem if not set to "yes" (which will be the hardcoded behavior in the future).

This option controls whether the netlogon server (currently only in 'active directory domain controller' mode), will reject clients which does not support NETLOGON_NEG_SUPPORTS_AES.

Support for NETLOGON_NEG_SUPPORTS_AES was added in Windows starting with Server 2008R2 and Windows 7, it's available in Samba starting with 4.0, however third party domain members like NetApp ONTAP still uses RC4 (HMAC-MD5), see https://www.samba.org/samba/security/CVE-2022-38023.html for more details.

The default changed from 'no' to 'yes', with the patches for CVE-2022-38023 see https://bugzilla.samba.org/show_bug.cgi?id=15240.

Avoid using this option! Use an explicit per machine account 'server reject md5 schannel:COMPUTERACCOUNT' instead! Which is available with the patches for CVE-2022-38023 see https://bugzilla.samba.org/show_bug.cgi?id=15240.

Samba will log an error in the log files at log level 0 if legacy a client is rejected or allowed without an explicit, 'server reject md5 schannel:COMPUTERACCOUNT = no' option for the client. The message will indicate the explicit 'server reject md5 schannel:COMPUTERACCOUNT = no' line to be added, if the legacy client software requires it. (The log level can be adjusted with 'CVE_2022_38023:error_debug_level = 1' in order to complain only at a higher log level).

This allows admins to use "no" only for a short grace period, in order to collect the explicit 'server reject md5 schannel:COMPUTERACCOUNT = no' options.

When set to 'yes' this option overrides the 'allow nt4 crypto:COMPUTERACCOUNT' and 'allow nt4 crypto' options and implies 'allow nt4 crypto:COMPUTERACCOUNT = no'.

Default: reject md5 clients = yes

server reject md5 schannel:COMPUTERACCOUNT (G)

If you still have legacy domain members or trusted domains, which required "reject md5 clients = no" before, it is possible to specify an explicit exception per computer account by setting 'server reject md5 schannel:COMPUTERACCOUNT = no'. Note that COMPUTERACCOUNT has to be the sAMAccountName value of the computer account (including the trailing '$' sign).

Samba will log a complaint in the log files at log level 0 about the security problem if the option is set to "no", but the related computer does not require it. (The log level can be adjusted with

'CVE_2022_38023:warn_about_unused_debug_level = 1' in order to

complain only at a higher log level).

Samba will log a warning in the log files at log level 5 if a

setting is still needed for the specified computer account.

See CVE-2022-38023,

https://bugzilla.samba.org/show_bug.cgi?id=15240.

This option overrides the reject md5 clients option.

When set to 'yes' this option overrides the 'allow nt4

crypto:COMPUTERACCOUNT' and 'allow nt4 crypto' options and implies

'allow nt4 crypto:COMPUTERACCOUNT = no'.

```
server reject md5 schannel:LEGACYCOMPUTER1$ = no

server reject md5 schannel:NASBOX$ = no

server reject md5 schannel:LEGACYCOMPUTER2$ = no
```

No default

reject md5 servers (G)

This option controls whether winbindd requires support for aes

support for the netlogon secure channel.

The following flags will be required NETLOGON_NEG_ARCFOUR,

NETLOGON_NEG_SUPPORTS_AES, NETLOGON_NEG_PASSWORD_SET2 and

NETLOGON_NEG_AUTHENTICATED_RPC.

You can set this to yes if all domain controllers support aes. This

will prevent downgrade attacks.

The behavior can be controlled per netbios domain by using 'reject

md5 servers:NETBIOSDOMAIN = no' as option.

The default changed from 'no' to 'yes, with the patches for

CVE-2022-38023, see

https://bugzilla.samba.org/show_bug.cgi?id=15240

This option overrides the require strong key option.

Default: reject md5 servers = yes

remote announce (G)

This option allows you to setup nmbd(8) to periodically announce

itself to arbitrary IP addresses with an arbitrary workgroup name.

This is useful if you want your Samba server to appear in a remote workgroup for which the normal browse propagation rules don't work. The remote workgroup can be anywhere that you can send IP packets to.

For example:

```
    remote announce = 192.168.2.255/SERVERS 192.168.4.255/STAFF
```

the above line would cause nmbd to announce itself to the two given IP addresses using the given workgroup names. If you leave out the workgroup name, then the one given in the workgroup parameter is used instead.

The IP addresses you choose would normally be the broadcast addresses of the remote networks, but can also be the IP addresses of known browse masters if your network config is that stable.

See the chapter on Network Browsing in the Samba-HOWTO book.

Default: remote announce =

remote browse sync (G)

This option allows you to setup nmbd(8) to periodically request synchronization of browse lists with the master browser of a Samba server that is on a remote segment. This option will allow you to gain browse lists for multiple workgroups across routed networks. This is done in a manner that does not work with any non-Samba servers.

This is useful if you want your Samba server and all local clients to appear in a remote workgroup for which the normal browse propagation rules don't work. The remote workgroup can be anywhere that you can send IP packets to.

For example:

```
    remote browse sync = 192.168.2.255 192.168.4.255
```

the above line would cause nmbd to request the master browser on the specified subnets or addresses to synchronize their browse lists with the local server.

The IP addresses you choose would normally be the broadcast addresses of the remote networks, but can also be the IP addresses of known browse masters if your network config is that stable. If a machine IP address is given Samba makes NO attempt to validate that the remote machine is available, is listening, nor that it is in fact the browse master on its segment.

The remote browse sync may be used on networks where there is no WINS server, and may be used on disjoint networks where each network has its own WINS server.

Default: remote browse sync =

rename user script (G)

This is the full pathname to a script that will be run as root by smbd(8) under special circumstances described below.

When a user with admin authority or SeAddUserPrivilege rights renames a user (e.g.: from the NT4 User Manager for Domains), this script will be run to rename the POSIX user. Two variables, %uold and %unew, will be substituted with the old and new usernames, respectively. The script should return 0 upon successful completion, and nonzero otherwise.

Note

The script has all responsibility to rename all the necessary data that is accessible in this posix method. This can mean different requirements for different backends. The tdbsam and smbpasswd backends will take care of the contents of their respective files, so the script is responsible only for changing the POSIX username, and other data that may required for your circumstances, such as home directory. Please also consider whether or not you need to rename the actual home directories themselves. The ldapsam backend will not make any changes, because of the potential issues with renaming the LDAP naming attribute. In this case the script is responsible for changing the attribute that samba uses (uid) for locating users, as well as any data that needs to change for other applications using the same directory.

Default: rename user script =

require strong key (G)

This option controls whether winbindd requires support for md5 strong key support for the netlogon secure channel.

The following flags will be required NETLOGON_NEG_STRONG_KEYS, NETLOGON_NEG_ARCFOUR and NETLOGON_NEG_AUTHENTICATED_RPC.

You can set this to no if some domain controllers only support des. This might allows weak crypto to be negotiated, may via downgrade attacks.

The behavior can be controlled per netbios domain by using 'require strong key:NETBIOSDOMAIN = no' as option.

Note for active directory domain this option is hardcoded to 'yes'

This option is over-ridden by the reject md5 servers option.

This option overrides the client schannel option.

Default: require strong key = yes

reset on zero vc (G)

This boolean option controls whether an incoming SMB1 session setup should kill other connections coming from the same IP. This matches the default Windows 2003 behaviour. Setting this parameter to yes becomes necessary when you have a flaky network and windows decides to reconnect while the old connection still has files with share modes open. These files become inaccessible over the new connection. The client sends a zero VC on the new connection, and Windows 2003 kills all other connections coming from the same IP. This way the locked files are accessible again. Please be aware that enabling this option will kill connections behind a masquerading router, and will not trigger for clients that only use SMB2 or SMB3.

Default: reset on zero vc = no

restrict anonymous (G)

The setting of this parameter determines whether SAMR and LSA DCERPC services can be accessed anonymously. This corresponds to the following Windows Server registry options:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa\RestrictAnonymous

The option also affects the browse option which is required by legacy clients which rely on Netbios browsing. While modern Windows version should be fine with restricting the access there could still be applications relying on anonymous access.

Setting restrict anonymous = 1 will disable anonymous SAMR access.

Setting restrict anonymous = 2 will, in addition to restricting SAMR access, disallow anonymous connections to the IPC$ share in general. Setting guest ok = yes on any share will remove the security advantage.

Default: restrict anonymous = 0

root

This parameter is a synonym for root directory.

root dir

This parameter is a synonym for root directory.

root directory (G)

The server will chroot() (i.e. Change its root directory) to this directory on startup. This is not strictly necessary for secure operation. Even without it the server will deny access to files not in one of the service entries. It may also check for, and deny access to, soft links to other parts of the filesystem, or attempts to use ".." in file names to access other directories (depending on the setting of the wide smbconfoptions parameter).

Adding a root directory entry other than "/" adds an extra level of security, but at a price. It absolutely ensures that no access is given to files not in the sub-tree specified in the root directory option, including some files needed for complete operation of the server. To maintain full operability of the server you will need to mirror some system files into the root directory tree. In particular you will need to mirror /etc/passwd (or a subset of it), and any binaries or configuration files needed for printing (if required). The set of files that must be mirrored is operating system dependent.

Default: root directory =

Example: root directory = /homes/smb

root postexec (S)

This is the same as the postexec parameter except that the command is run as root. This is useful for unmounting filesystems (such as CDROMs) after a connection is closed.

Default: root postexec =

root preexec (S)

This is the same as the preexec parameter except that the command is run as root. This is useful for mounting filesystems (such as CDROMs) when a connection is opened.

Default: root preexec =

root preexec close (S)

This is the same as the preexec close parameter except that the command is run as root.

Default: root preexec close = no

rpc big endian (G)

Setting this option will force the RPC client and server to transfer data in big endian.

If it is disabled, data will be transferred in little endian.

The behaviour is independent of the endianness of the host machine.

Default: rpc big endian = no

rpc server dynamic port range (G)

This parameter tells the RPC server which port range it is allowed to use to create a listening socket for LSA, SAM, Netlogon and others without wellknown tcp ports. The first value is the lowest number of the port range and the second the highest.

This applies to RPC servers in all server roles.

Default: rpc server dynamic port range = 49152-65535

rpc server port (G)

Specifies which port the server should listen on for DCE/RPC over TCP/IP traffic.

This controls the default port for all protocols, except for NETLOGON.

If unset, the first available port from rpc server dynamic port range is used, e.g. 49152.

The NETLOGON server will use the next available port, e.g. 49153.

To change this port use (eg) rpc server port:netlogon = 4000.

Furthermore, all RPC servers can have the port they use specified independenty, with (for example) rpc server port:drsuapi = 5000.

This option applies currently only when samba(8) runs as an active directory domain controller.

The default value 0 causes Samba to select the first available port from rpc server dynamic port range.

Default: rpc server port = 0

rpc start on demand helpers (G)

This global parameter determines if samba-dcerpcd should be started on demand to service named pipe (np) DCE-RPC requests from smbd or winbindd. This is the normal case where no startup scripts have been modified to start samba-dcerpcd as a daemon.

If samba-dcerpcd is started as a daemon or via a system service manager such as systemd, this parameter MUST be set to "no", otherwise samba-dcerpcd will fail to start.

Default: rpc start on demand helpers = yes

samba kcc command (G)

This option specifies the path to the Samba KCC command. This script is used for replication topology replication.

It should not be necessary to modify this option except for testing purposes or if the samba_kcc was installed in a non-default location.

Default: samba kcc command = /builddir/build/BUILD/samba-4.17.5/source4/scripting/bin/samba_kcc

Example: samba kcc command = /usr/local/bin/kcc

security (G)

This option affects how clients respond to Samba and is one of the most important settings in the smb.conf file.

The default is security = user, as this is the most common setting, used for a standalone file server or a DC.

The alternatives are security = ads or security = domain, which support joining Samba to a Windows domain

You should use security = user and map to guest if you want to mainly setup shares without a password (guest shares). This is commonly used for a shared printer server.

The different settings will now be explained.

SECURITY = AUTO

This is the default security setting in Samba, and causes Samba to consult the server role parameter (if set) to determine the security mode.

SECURITY = USER

If server role is not specified, this is the default security setting in Samba. With user-level security a client must first "log-on" with a valid username and password (which can be mapped using the username map parameter). Encrypted passwords (see the encrypted passwords parameter) can also be used in this security mode. Parameters such as user and guest only if set are then applied and may change the UNIX user to use on this connection, but only after the user has been successfully authenticated.

Note that the name of the resource being requested is not sent to the server until after the server has successfully authenticated the client. This is why guest shares don't work in user level security without allowing the server to automatically map unknown users into the guest account. See the map to guest parameter for details on doing this.

SECURITY = DOMAIN

This mode will only work correctly if net(8) has been used to add this machine into a Windows NT Domain. It expects the encrypted passwords parameter to be set to yes. In this mode Samba will try to validate the username/password by passing it to a Windows NT Primary or Backup Domain Controller, in exactly the same way that a Windows NT Server would do.

Note that a valid UNIX user must still exist as well as the account on the Domain Controller to allow Samba to have a valid UNIX account to map file access to.

Note that from the client's point of view security = domain is the same as security = user. It only affects how the server deals with the authentication, it does not in any way affect what the client sees.

Note that the name of the resource being requested is not sent to the server until after the server has successfully authenticated the client. This is why guest shares don't work in user level security without allowing the server to automatically map unknown users into the guest account. See the map to guest parameter for details on doing this.

See also the password server parameter and the encrypted passwords parameter.

SECURITY = ADS

In this mode, Samba will act as a domain member in an ADS realm. To operate in this mode, the machine running Samba will need to have Kerberos installed and configured and Samba will need to be joined to the ADS realm using the net utility.

Note that this mode does NOT make Samba operate as a Active Directory Domain Controller.

Note that this forces require strong key = yes and client schannel = yes for the primary domain.

Read the chapter about Domain Membership in the HOWTO for details.

Default: security = AUTO

Example: security = DOMAIN

security mask (S)

This parameter has been removed for Samba 4.0.0.

No default

max protocol

This parameter is a synonym for server max protocol.

protocol

This parameter is a synonym for server max protocol.

server max protocol (G)

The value of the parameter (a string) is the highest protocol level that will be supported by the server.

Possible values are :

- ? LANMAN1: First modern version of the protocol. Long filename support.
- ? LANMAN2: Updates to Lanman1 protocol.
- ? NT1: Current up to date version of the protocol. Used by Windows NT. Known as CIFS.
- ? SMB2: Re-implementation of the SMB protocol. Used by Windows Vista and later versions of Windows. SMB2 has sub protocols available.
  - ? SMB2_02: The earliest SMB2 version.
  - ? SMB2_10: Windows 7 SMB2 version.

  By default SMB2 selects the SMB2_10 variant.
- ? SMB3: The same as SMB2. Used by Windows 8. SMB3 has sub protocols available.
  - ? SMB3_00: Windows 8 SMB3 version.
  - ? SMB3_02: Windows 8.1 SMB3 version.
  - ? SMB3_11: Windows 10 SMB3 version.

  By default SMB3 selects the SMB3_11 variant.

Normally this option should not be set as the automatic negotiation phase in the SMB protocol takes care of choosing the appropriate protocol.

Default: server max protocol = SMB3

Example: server max protocol = LANMAN1

min protocol

This parameter is a synonym for server min protocol.

server min protocol (G)

This setting controls the minimum protocol version that the server will allow the client to use.

Normally this option should not be set as the automatic negotiation phase in the SMB protocol takes care of choosing the appropriate protocol unless you have legacy clients which are SMB1 capable only.

See Related command: server max protocol for a full list of available protocols.

Default: server min protocol = SMB2_02

Example: server min protocol = NT1

server multi channel support (G)

This boolean parameter controls whether smbd(8) will support SMB3 multi-channel.

This parameter was added with version 4.4.

Note that this feature was still considered experimental up to 4.14.

Due to dependencies to kernel APIs of Linux or FreeBSD, it's only possible to use this feature on Linux and FreeBSD for now. For testing this restriction can be overwritten by specifying force:server multi channel support=yes in addition.

This option is enabled by default starting with to 4.15 (on Linux and FreeBSD).

Default: server multi channel support = yes

server role (G)

This option determines the basic operating mode of a Samba server and is one of the most important settings in the smb.conf file.

The default is server role = auto, as causes Samba to operate according to the security setting, or if not specified as a simple file server that is not connected to any domain.

The alternatives are server role = standalone or server role = member server, which support joining Samba to a Windows domain, along with server role = domain controller, which run Samba as a Windows domain controller.

You should use server role = standalone and map to guest if you want to mainly setup shares without a password (guest shares). This is commonly used for a shared printer server.

SERVER ROLE = AUTO

This is the default server role in Samba, and causes Samba to consult the security parameter (if set) to determine the server role, giving compatible behaviours to previous Samba versions.

SERVER ROLE = STANDALONE

If security is also not specified, this is the default security setting in Samba. In standalone operation, a client must first "log-on" with a valid username and password (which can be mapped using the username map parameter) stored on this machine. Encrypted passwords (see the encrypted passwords parameter) are by default used in this security mode. Parameters such as user and guest only if set are then applied and may change the UNIX user to use on this connection, but only after the user has been successfully authenticated.

SERVER ROLE = MEMBER SERVER

This mode will only work correctly if net(8) has been used to add this machine into a Windows Domain. It expects the encrypted passwords parameter to be set to yes. In this mode Samba will try to validate the username/password by passing it to a Windows or Samba Domain Controller, in exactly the same way that a Windows Server would do.

Note that a valid UNIX user must still exist as well as the account on the Domain Controller to allow Samba to have a valid UNIX account to map file access to. Winbind can provide this.

SERVER ROLE = CLASSIC PRIMARY DOMAIN CONTROLLER

This mode of operation runs a classic Samba primary domain controller, providing domain logon services to Windows and Samba clients of an NT4-like domain. Clients must be joined to the domain to create a secure, trusted path across the network. There must be only one PDC per NetBIOS scope (typcially a broadcast network or clients served by a single WINS server).

SERVER ROLE = CLASSIC BACKUP DOMAIN CONTROLLER

This mode of operation runs a classic Samba backup domain controller, providing domain logon services to Windows and Samba clients of an NT4-like domain. As a BDC, this allows multiple Samba servers to provide redundant logon services to a single NetBIOS scope.

SERVER ROLE = ACTIVE DIRECTORY DOMAIN CONTROLLER

This mode of operation runs Samba as an active directory domain controller, providing domain logon services to Windows and Samba clients of the domain. This role requires special configuration, see the Samba4 HOWTO

SERVER ROLE = IPA DOMAIN CONTROLLER

This mode of operation runs Samba in a hybrid mode for IPA domain controller, providing forest trust to Active Directory. This role requires special configuration performed by IPA installers and should not be used manually by any administrator.

Default: server role = AUTO

Example: server role = ACTIVE DIRECTORY DOMAIN CONTROLLER

server schannel (G)

This option is deprecated and will be removed in future, as it is a security problem if not set to "yes" (which will be the hardcoded behavior in future).

Avoid using this option! Use explicit 'server require schannel:COMPUTERACCOUNT = no' instead!

Samba will log an error in the log files at log level 0 if legacy a client is rejected or allowed without an explicit, 'server require schannel:COMPUTERACCOUNT = no' option for the client. The message will indicate the explicit 'server require schannel:COMPUTERACCOUNT = no' line to be added, if the legacy client software requires it. (The log level can be adjusted with 'CVE_2020_1472:error_debug_level = 1' in order to complain only at a higher log level).

This allows admins to use "auto" only for a short grace period, in order to collect the explicit 'server require schannel:COMPUTERACCOUNT = no' options.

See CVE-2020-1472(ZeroLogon),

https://bugzilla.samba.org/show_bug.cgi?id=14497.

This option is over-ridden by the server require

schannel:COMPUTERACCOUNT option.

This option is over-ridden by the effective value of 'yes' from the

'server schannel require seal:COMPUTERACCOUNT' and/or 'server

schannel require seal' options.

Default: server schannel = yes

server require schannel:COMPUTERACCOUNT (G)

If you still have legacy domain members, which required "server

schannel = auto" before, it is possible to specify explicit

exception per computer account by using 'server require

schannel:COMPUTERACCOUNT = no' as option. Note that COMPUTERACCOUNT

has to be the sAMAccountName value of the computer account

(including the trailing '$' sign).

Samba will complain in the log files at log level 0, about the

security problem if the option is not set to "no", but the related

computer is actually using the netlogon secure channel (schannel)

feature. (The log level can be adjusted with

'CVE_2020_1472:warn_about_unused_debug_level = 1' in order to

complain only at a higher log level).

Samba will warn in the log files at log level 5, if a setting is

still needed for the specified computer account.

See CVE-2020-1472(ZeroLogon),

https://bugzilla.samba.org/show_bug.cgi?id=14497.

This option overrides the server schannel option.

This option is over-ridden by the effective value of 'yes' from the

'server schannel require seal:COMPUTERACCOUNT' and/or 'server

schannel require seal' options.

Which means 'server require schannel:COMPUTERACCOUNT = no' is only

useful in combination with 'server schannel require

seal:COMPUTERACCOUNT = no'

```
server require schannel:LEGACYCOMPUTER1$ = no
```

```
        server require schannel seal:LEGACYCOMPUTER1$ = no
        server require schannel:NASBOX$ = no
        server require schannel seal:NASBOX$ = no
        server require schannel:LEGACYCOMPUTER2$ = no
        server require schannel seal:LEGACYCOMPUTER2$ = no
```

No default

server schannel require seal (G)

This option is deprecated and will be removed in future, as it is a security problem if not set to "yes" (which will be the hardcoded behavior in future).

This option controls whether the netlogon server, will reject the usage of netlogon secure channel without privacy/enryption.

The option is modelled after the registry key available on Windows.

```
        HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Netlogon\Parameters\RequireSeal=2
```

Avoid using this option! Use the per computer account specific option 'server schannel require seal:COMPUTERACCOUNT' instead! Which is available with the patches for CVE-2022-38023 see https://bugzilla.samba.org/show_bug.cgi?id=15240.

Samba will log an error in the log files at log level 0 if legacy a client is rejected or allowed without an explicit, 'server schannel require seal:COMPUTERACCOUNT = no' option for the client. The message will indicate the explicit 'server schannel require seal:COMPUTERACCOUNT = no' line to be added, if the legacy client software requires it. (The log level can be adjusted with 'CVE_2022_38023:error_debug_level = 1' in order to complain only at a higher log level).

This allows admins to use "no" only for a short grace period, in order to collect the explicit 'server schannel require seal:COMPUTERACCOUNT = no' options.

When set to 'yes' this option overrides the 'server require schannel:COMPUTERACCOUNT' and 'server schannel' options and implies 'server require schannel:COMPUTERACCOUNT = yes'.

This option is over-ridden by the server schannel require seal:COMPUTERACCOUNT option.

Default: server schannel require seal = yes

server schannel require seal:COMPUTERACCOUNT (G)

If you still have legacy domain members, which required "server schannel require seal = no" before, it is possible to specify explicit exception per computer account by using 'server schannel require seal:COMPUTERACCOUNT = no' as option. Note that COMPUTERACCOUNT has to be the sAMAccountName value of the computer account (including the trailing '$' sign).

Samba will log a complaint in the log files at log level 0 about the security problem if the option is set to "no", but the related computer does not require it. (The log level can be adjusted with 'CVE_2022_38023:warn_about_unused_debug_level = 1' in order to complain only at a higher log level).

Samba will warn in the log files at log level 5, if a setting is still needed for the specified computer account.

See CVE-2022-38023, https://bugzilla.samba.org/show_bug.cgi?id=15240.

This option overrides the 'server schannel require seal' option.

When set to 'yes' this option overrides the 'server require schannel:COMPUTERACCOUNT' and 'server schannel' options and implies 'server require schannel:COMPUTERACCOUNT = yes'.

```
server require schannel seal:LEGACYCOMPUTER1$ = no
server require schannel seal:NASBOX$ = no
server require schannel seal:LEGACYCOMPUTER2$ = no
```

No default

server services (G)

This option contains the services that the Samba daemon will run.

An entry in the smb.conf file can either override the previous value completely or entries can be removed from or added to it by prefixing them with + or -.

Default: server services = s3fs, rpc, nbt, wrepl, ldap, cldap, kdc, drepl, winbindd, ntp_signd, kcc, dnsupdate, dns

Example: server services = -s3fs, +smb

server signing (G)

This controls whether the client is allowed or required to use SMB1 and SMB2 signing. Possible values are default, auto, mandatory and disabled.

By default, and when smb signing is set to default, smb signing is required when server role is active directory domain controller and disabled otherwise.

When set to auto, SMB1 signing is offered, but not enforced. When set to mandatory, SMB1 signing is required and if set to disabled, SMB signing is not offered either.

For the SMB2 protocol, by design, signing cannot be disabled. In the case where SMB2 is negotiated, if this parameter is set to disabled, it will be treated as auto. Setting it to mandatory will still require SMB2 clients to use signing.

Default: server signing = default

server smb encrypt (S)

This parameter controls whether a remote client is allowed or required to use SMB encryption. It has different effects depending on whether the connection uses SMB1 or SMB2 and newer:

? If the connection uses SMB1, then this option controls the use of a Samba-specific extension to the SMB protocol introduced in Samba 3.2 that makes use of the Unix extensions.

? If the connection uses SMB2 or newer, then this option controls the use of the SMB-level encryption that is supported in SMB version 3.0 and above and available in Windows 8 and newer.

This parameter can be set globally and on a per-share bases. Possible values are off, if_required, desired, and required. A special value is default which is the implicit default setting of if_required.

Effects for SMB1

The Samba-specific encryption of SMB1 connections is an extension to the SMB protocol negotiated as part of the UNIX extensions. SMB encryption uses the GSSAPI (SSPI on Windows) ability to encrypt and sign every request/response in a SMB protocol stream. When enabled it provides a secure method of SMB/CIFS communication, similar to an ssh protected session, but using SMB/CIFS authentication to negotiate encryption and signing keys. Currently this is only supported smbclient of by Samba 3.2 and newer, and hopefully soon Linux CIFSFS and MacOS/X clients. Windows clients do not support this feature.

This may be set on a per-share basis, but clients may chose to encrypt the entire session, not just traffic to a specific share. If this is set to mandatory then all traffic to a share must be encrypted once the connection has been made to the share. The server would return "access denied" to all non-encrypted requests on such a share. Selecting encrypted traffic reduces throughput as smaller packet sizes must be used (no huge UNIX style read/writes allowed) as well as the overhead of encrypting and signing all the data.

If SMB encryption is selected, Windows style SMB signing (see the server signing option) is no longer necessary, as the GSSAPI flags use select both signing and sealing of the data.

When set to auto or default, SMB encryption is offered, but not enforced. When set to mandatory, SMB encryption is required and if set to disabled, SMB encryption can not be negotiated.

Effects for SMB2 and newer

Native SMB transport encryption is available in SMB version 3.0 or newer. It is only offered by Samba if server max protocol is set to SMB3 or newer. Clients supporting this type of encryption include Windows 8 and newer, Windows server 2012 and newer, and smbclient of Samba 4.1 and newer.

The protocol implementation offers various options:

? The capability to perform SMB encryption can be negotiated during protocol negotiation.

? Data encryption can be enabled globally. In that case, an encryption-capable connection will have all traffic in all its sessions encrypted. In particular all share connections will be encrypted.

? Data encryption can also be enabled per share if not enabled globally. For an encryption-capable connection, all connections to an encryption-enabled share will be encrypted.

? Encryption can be enforced. This means that session setups will be denied on non-encryption-capable connections if data encryption has been enabled globally. And tree connections will be denied for non-encryption capable connections to shares with data encryption enabled.

These features can be controlled with settings of server smb encrypt as follows:

? Leaving it as default, explicitly setting default, or setting it to if_required globally will enable negotiation of encryption but will not turn on data encryption globally or per share.

? Setting it to desired globally will enable negotiation and will turn on data encryption on sessions and share connections for those clients that support it.

? Setting it to required globally will enable negotiation and turn on data encryption on sessions and share connections. Clients that do not support encryption will be denied access to the server.

? Setting it to off globally will completely disable the encryption feature for all connections. Setting server smb encrypt = required for individual shares (while it's globally off) will deny access to this shares for all clients.

? Setting it to desired on a share will turn on data encryption for this share for clients that support encryption if negotiation has been enabled globally.

? Setting it to required on a share will enforce data encryption for this share if negotiation has been enabled globally. I.e. clients that do not support encryption will be denied access to the share.

Note that this allows per-share enforcing to be controlled in Samba differently from Windows: In Windows, RejectUnencryptedAccess is a global setting, and if it is set, all shares with data encryption turned on are automatically enforcing encryption. In order to achieve the same effect in Samba, one has to globally set server smb encrypt to if_required, and then set all shares that should be encrypted to required. Additionally, it is possible in Samba to have some shares with encryption required and some other shares with encryption only desired, which is not possible in Windows.

? Setting it to off or if_required for a share has no effect.

Default: server smb encrypt = default

server smb3 encryption algorithms (G)

This parameter specifies the availability and order of encryption algorithms which are available for negotiation in the SMB3_11 dialect.

It is also possible to remove individual algorithms from the default list, by prefixing them with '-'. This can avoid having to specify a hardcoded list.

Note: that the removal of AES-128-CCM from the list will result in SMB3_00 and SMB3_02 being unavailable, as it is the default and only available algorithm for these dialects.

Default: server smb3 encryption algorithms = AES-128-GCM,

AES-128-CCM, AES-256-GCM, AES-256-CCM

Example: server smb3 encryption algorithms = AES-256-GCM

Example: server smb3 encryption algorithms = -AES-128-GCM

-AES-128-CCM

server smb3 signing algorithms (G)

This parameter specifies the availability and order of signing

algorithms which are available for negotiation in the SMB3_11

dialect.

It is also possible to remove individual algorithms from the

default list, by prefixing them with '-'. This can avoid having to

specify a hardcoded list.

Note: that the removal of AES-128-CMAC from the list will result in

SMB3_00 and SMB3_02 being unavailable, and the removal of

HMAC-SHA256 will result in SMB2_02 and SMB2_10 being unavailable,

as these are the default and only available algorithms for these

dialects.

Default: server smb3 signing algorithms = AES-128-GMAC,

AES-128-CMAC, HMAC-SHA256

Example: server smb3 signing algorithms = AES-128-CMAC, HMAC-SHA256

Example: server smb3 signing algorithms = -AES-128-CMAC

server string (G)

This controls what string will show up in the printer comment box

in print manager and next to the IPC connection in net view. It can

be any string that you wish to show to your users.

It also sets what will appear in browse lists next to the machine

name.

A %v will be replaced with the Samba version number.

A %h will be replaced with the hostname.

Default: server string = Samba %v

Example: server string = University of GNUs Samba Server

set primary group script (G)

Thanks to the Posix subsystem in NT a Windows User has a primary group in addition to the auxiliary groups. This script sets the primary group in the unix user database when an administrator sets the primary group from the windows user manager or when fetching a SAM with net rpc vampire. %u will be replaced with the user whose primary group is to be set. %g will be replaced with the group to set.

Default: set primary group script =

Example: set primary group script = /usr/sbin/usermod -g '%g' '%u'

set quota command (G)

The set quota command should only be used whenever there is no operating system API available from the OS that samba can use.

This option is only available if Samba was compiled with quota support.

This parameter should specify the path to a script that can set quota for the specified arguments.

The specified script should take the following arguments:

? 1 - path to where the quota needs to be set. This needs to be interpreted relative to the current working directory that the script may also check for.

? 2 - quota type

    ? 1 - user quotas

    ? 2 - user default quotas (uid = -1)

    ? 3 - group quotas

    ? 4 - group default quotas (gid = -1)

? 3 - id (uid for user, gid for group, -1 if N/A)

? 4 - quota state (0 = disable, 1 = enable, 2 = enable and enforce)

? 5 - block softlimit

? 6 - block hardlimit

? 7 - inode softlimit

? 8 - inode hardlimit

? 9(optional) - block size, defaults to 1024

The script should output at least one line of data on success. And nothing on failure.

Default: set quota command =

Example: set quota command = /usr/local/sbin/set_quota

share:fake_fscaps (G)

This is needed to support some special application that makes QFSINFO calls to check whether we set the SPARSE_FILES bit (0x40). If this bit is not set that particular application refuses to work against Samba. With share:fake_fscaps = 64 the SPARSE_FILES file system capability flag is set. Use other decimal values to specify the bitmask you need to fake.

Default: share:fake_fscaps = 0

short preserve case (S)

This boolean parameter controls if new files which conform to 8.3 syntax, that is all in upper case and of suitable length, are created upper case, or if they are forced to be the default case. This option can be use with preserve case = yes to permit long filenames to retain their case, while short names are lowered.

See the section on NAME MANGLING.

Default: short preserve case = yes

show add printer wizard (G)

With the introduction of MS-RPC based printing support for Windows NT/2000 client in Samba 2.2, a "Printers..." folder will appear on Samba hosts in the share listing. Normally this folder will contain an icon for the MS Add Printer Wizard (APW). However, it is possible to disable this feature regardless of the level of privilege of the connected user.

Under normal circumstances, the Windows NT/2000 client will open a handle on the printer server with OpenPrinterEx() asking for Administrator privileges. If the user does not have administrative access on the print server (i.e is not root or has granted the SePrintOperatorPrivilege), the OpenPrinterEx() call fails and the client makes another open call with a request for a lower privilege level. This should succeed, however the APW icon will not be displayed.

Disabling the show add printer wizard parameter will always cause the OpenPrinterEx() on the server to fail. Thus the APW icon will never be displayed.

Note

This does not prevent the same user from having administrative privilege on an individual printer.

Default: show add printer wizard = yes

shutdown script (G)

This a full path name to a script called by smbd(8) that should start a shutdown procedure.

If the connected user possesses the SeRemoteShutdownPrivilege, right, this command will be run as root.

The %z %t %r %f variables are expanded as follows:

? %z will be substituted with the shutdown message sent to the server.

? %t will be substituted with the number of seconds to wait before effectively starting the shutdown procedure.

? %r will be substituted with the switch -r. It means reboot after shutdown for NT.

? %f will be substituted with the switch -f. It means force the shutdown even if applications do not respond for NT.

Shutdown script example:

```
#!/bin/bash
time=$2
let time="${time} / 60"
let time="${time} + 1"
/sbin/shutdown $3 $4 +$time $1 &
```

Shutdown does not return so we need to launch it in background.

Default: shutdown script =

Example: shutdown script = /usr/local/samba/sbin/shutdown %m %t %r %f unix extensions

This parameter is a synonym for smb1 unix extensions.

smb1 unix extensions (G)

This boolean parameter controls whether Samba implements the SMB1/CIFS UNIX extensions, as defined by HP. These extensions enable Samba to better serve UNIX SMB1/CIFS clients by supporting features such as symbolic links, hard links, etc... These extensions require a similarly enabled client, and are of no current use to Windows clients.

Note if this parameter is turned on, the wide links parameter will automatically be disabled.

See the parameter allow insecure wide links if you wish to change this coupling between the two parameters.

Default: smb1 unix extensions = yes

smb2 disable lock sequence checking (G)

This boolean parameter controls whether smbd(8) will disable lock sequence checking even for multi-channel connections as well as durable handles.

The [MS-SMB2] specification (under 3.3.5.14 Receiving an SMB2 LOCK Request) documents that a server should do lock sequence if Open.IsResilient or Open.IsDurable or Open.IsPersistent is TRUE or if Connection.Dialect belongs to the SMB 3.x dialect family and Connection.ServerCapabilities includes SMB2_GLOBAL_CAP_MULTI_CHANNEL.

But Windows Server (at least up to v2004) only does these checks for the Open.IsResilient and Open.IsPersistent. That means they do not implement the behavior specified in [MS-SMB2].

By default Samba behaves according to the specification and implements lock sequence checking when multi-channel is used.

Warning: Only enable this option if existing clients can't handle lock sequence checking for handles without Open.IsResilient and Open.IsPersistent. And it turns out that the Windows Server behavior is required.

Note: it's likely that this option will be removed again if future Windows versions change their behavior.

Note: Samba does not implement Open.IsResilient and Open.IsPersistent yet.

Default: smb2 disable lock sequence checking = no

Example: smb2 disable lock sequence checking = yes

smb2 disable oplock break retry (G)

This boolean parameter controls whether smbd(8) will trigger smb2 oplock break notification retries when using server multi channel support = yes.

The [MS-SMB2] specification documents that a server should send smb2 oplock break notification retries on all available channel to the given client.

But Windows Server versions (at least up to 2019) do not send smb2 oplock break notification retries on channel failures. That means they do not implement the behavior specified in [MS-SMB2].

By default Samba behaves according to the specification and send smb2 oplock break notification retries.

Warning: Only enable this option if existing clients can't handle possible retries and it turns out that the Windows Server behavior is required.

Note: it's likely that this option gets removed again if future Windows versions change their behavior.

Note: this only applies to oplocks and not SMB2 leases.

Default: smb2 disable oplock break retry = no

Example: smb2 disable oplock break retry = yes

smb2 leases (G)

This boolean option tells smbd whether to globally negotiate SMB2 leases on file open requests. Leasing is an SMB2-only feature which allows clients to aggressively cache files locally above and beyond the caching allowed by SMB1 oplocks.

This is only available with oplocks = yes and kernel oplocks = no.

Note that the write cache won't be used for file handles with a smb2 write lease.

Default: smb2 leases = yes

smb2 max credits (G)

This option controls the maximum number of outstanding simultaneous SMB2 operations that Samba tells the client it will allow. This is similar to the max mux parameter for SMB1. You should never need to set this parameter.

The default is 8192 credits, which is the same as a Windows 2008R2 SMB2 server.

Default: smb2 max credits = 8192

smb2 max read (G)

This option specifies the protocol value that smbd(8) will return to a client, informing the client of the largest size that may be returned by a single SMB2 read call.

The maximum is 8388608 bytes (8MiB), which is the same as a Windows Server 2012 r2.

Please note that the default is 8MiB, but it's limit is based on the smb2 dialect (64KiB for SMB == 2.0, 8MiB for SMB >= 2.1 with LargeMTU). Large MTU is not supported over NBT (tcp port 139).

Default: smb2 max read = 8388608

smb2 max trans (G)

This option specifies the protocol value that smbd(8) will return to a client, informing the client of the largest size of buffer that may be used in querying file meta-data via QUERY_INFO and related SMB2 calls.

The maximum is 8388608 bytes (8MiB), which is the same as a Windows Server 2012 r2.

Please note that the default is 8MiB, but it's limit is based on the smb2 dialect (64KiB for SMB == 2.0, 1MiB for SMB >= 2.1 with LargeMTU). Large MTU is not supported over NBT (tcp port 139).

Default: smb2 max trans = 8388608

smb2 max write (G)

This option specifies the protocol value that smbd(8) will return to a client, informing the client of the largest size that may be sent to the server by a single SMB2 write call.

The maximum is 8388608 bytes (8MiB), which is the same as a Windows Server 2012 r2.

Please note that the default is 8MiB, but it's limit is based on the smb2 dialect (64KiB for SMB == 2.0, 8MiB for SMB => 2.1 with LargeMTU). Large MTU is not supported over NBT (tcp port 139).

Default: smb2 max write = 8388608

smbd async dosmode (S)

This parameter control whether the fileserver will use sync or async methods for fetching the DOS attributes when doing a directory listing. By default sync methods will be used.

Default: smbd async dosmode = no

smbd getinfo ask sharemode (S)

This parameter allows disabling fetching file write time from the open file handle database locking.tdb when a client requests file or directory metadata. It's a performance optimisation at the expense of protocol correctness.

Default: smbd getinfo ask sharemode = yes

smbd max async dosmode (S)

This parameter controls how many async operations to fetch the DOS attributes the fileserver will queue when doing directory listings.

Default: smbd max async dosmode = aio max threads * 2

smbd max xattr size (S)

This parameter controls the maximum size of extended attributes that may be written to the server as EAs or as alternate data streams if vfs_streams_xattr is enabled. The maximum size of extended attributes depends on the Samba server's operating system and the underlying filesystem. The Linux VFS currently sets an upper boundary of 64 KiB per extended attribute. FreeBSD does not set a practical upper limit, but since pread() and pwrite() are not possible via the extattr on FreeBSD, it is not recommended to increase this value above a few MiB. If a client attempts to write an overly-large alternate datastream, the Samba server will return STATUS_FILESYSTEM_LIMITATION. If this error is encountered, users may try increasing the maximum size supported for xattr writes. If this is not possible, and writes are from a MacOS client and to an AFP_Resource extended attribute, the user may enable the vfs_fruit module and configure to allow stream writes for AFP_Resource to an alternative storage location. See vfs_fruit documentation for further details.

Default: smbd max xattr size = 65536

smbd profiling level (G)

This parameter allows the administrator to enable profiling support.

Possible values are off, count and on.

Default: smbd profiling level = off

Example: smbd profiling level = on

smbd search ask sharemode (S)

This parameter allows disabling fetching file write time from the open file handle database locking.tdb. It's a performance optimisation at the expense of protocol correctness.

Default: smbd search ask sharemode = yes

smb encrypt (S)

This is a synonym for server smb encrypt.

Default: smb encrypt = default

smb passwd file (G)

This option sets the path to the encrypted smbpasswd file. By default the path to the smbpasswd file is compiled into Samba.

An example of use is:

```
smb passwd file = /etc/samba/smbpasswd
```

Default: smb passwd file = /var/lib/samba/private/smbpasswd

smb ports (G)

Specifies which ports the server should listen on for SMB traffic.

Default: smb ports = 445 139

socket options (G)

> Warning
>
> Modern server operating systems are tuned for high network performance in the majority of situations; when you set socket options you are overriding those settings. Linux in particular has an auto-tuning mechanism for buffer sizes that will be disabled if you specify a socket buffer size. This can potentially cripple your TCP/IP stack.
>
> Getting the socket options correct can make a big difference to your performance, but getting them wrong can degrade it by just as much. As with any other low level setting, if you must make changes to it, make small changes and test the effect before making any large changes.

This option allows you to set socket options to be used when talking with the client.

Socket options are controls on the networking layer of the operating systems which allow the connection to be tuned.

This option will typically be used to tune your Samba server for optimal performance for your local network. There is no way that Samba can know what the optimal parameters are for your net, so you must experiment and choose them yourself. We strongly suggest you read the appropriate documentation for your operating system first (perhaps man setsockopt will help).

You may find that on some systems Samba will say "Unknown socket option" when you supply an option. This means you either incorrectly typed it or you need to add an include file to includes.h for your OS. If the latter is the case please send the patch to samba-technical@lists.samba.org.

Any of the supported socket options may be combined in any way you like, as long as your OS allows it.

This is the list of socket options currently settable using this option:

? SO_KEEPALIVE

? SO_REUSEADDR

? SO_BROADCAST

? TCP_NODELAY

? TCP_KEEPCNT *

? TCP_KEEPIDLE *

? TCP_KEEPINTVL *

? IPTOS_LOWDELAY

? IPTOS_THROUGHPUT

? SO_REUSEPORT

? SO_SNDBUF *

? SO_RCVBUF *

? SO_SNDLOWAT *

? SO_RCVLOWAT *

? SO_SNDTIMEO *

? SO_RCVTIMEO *

? TCP_FASTACK *

? TCP_QUICKACK

? TCP_NODELAYACK

? TCP_KEEPALIVE_THRESHOLD *

? TCP_KEEPALIVE_ABORT_THRESHOLD *

? TCP_DEFER_ACCEPT *

? TCP_USER_TIMEOUT *

Those marked with a '*' take an integer argument. The others can optionally take a 1 or 0 argument to enable or disable the option, by default they will be enabled if you don't specify 1 or 0.

To specify an argument use the syntax SOME_OPTION = VALUE for example SO_SNDBUF = 8192. Note that you must not have any spaces before or after the = sign.

If you are on a local network then a sensible option might be:

socket options = IPTOS_LOWDELAY

If you have a local network then you could try:

socket options = IPTOS_LOWDELAY TCP_NODELAY

If you are on a wide area network then perhaps try setting IPTOS_THROUGHPUT.

Note that several of the options may cause your Samba server to fail completely. Use these options with caution!

Default: socket options = TCP_NODELAY

Example: socket options = IPTOS_LOWDELAY

spn update command (G)

This option sets the command that for updating servicePrincipalName names from spn_update_list.

Default: spn update command = /builddir/build/BUILD/samba-4.17.5/source4/scripting/bin/samba_spnupdate

Example: spn update command = /usr/local/sbin/spnupdate

spoolss: architecture (G)

Windows spoolss print clients only allow association of server-side drivers with printers when the driver architecture matches the advertised print server architecture. Samba's spoolss print server architecture can be changed using this parameter.

Default: spoolss: architecture = Windows x64

Example: spoolss: architecture = Windows NT x86

spoolss: os_major (G)

Windows might require a new os version number. This option allows to modify the build number. The complete default version number is: 5.0.2195 (Windows 2000). The example is 6.1.7601 (Windows 2008 R2).

Default: spoolss: os_major = 5

Example: spoolss: os_major = 6

spoolss: os_minor (G)

Windows might require a new os version number. This option allows to modify the build number. The complete default version number is: 5.0.2195 (Windows 2000). The example is 6.1.7601 (Windows 2008 R2).

Default: spoolss: os_minor = 0

Example: spoolss: os_minor = 1

spoolss: os_build (G)

Windows might require a new os version number. This option allows to modify the build number. The complete default version number is: 5.0.2195 (Windows 2000). The example is 6.1.7601 (Windows 2008 R2).

Default: spoolss: os_build = 2195

Example: spoolss: os_build = 7601

spoolss_client: os_major (G)

Windows might require a new os version number. This option allows to modify the build number. The complete default version number is: 6.1.7007 (Windows 7 and Windows Server 2008 R2).

Default: spoolss_client: os_major = 6

spoolss_client: os_minor (G)

Windows might require a new os version number. This option allows to modify the build number. The complete default version number is: 6.1.7007 (Windows 7 and Windows Server 2008 R2).

Default: spoolss_client: os_minor = 1

spoolss_client: os_build (G)

Windows might require a new os version number. This option allows to modify the build number. The complete default version number is: 6.1.7007 (Windows 7 and Windows Server 2008 R2).

Default: spoolss_client: os_build = 7007

spotlight (S)

This parameter controls whether Samba allows Spotlight queries on a share. For controlling indexing of filesystems you also have to use Tracker's own configuration system.

Spotlight has several prerequisites:

? Samba must be configured and built with Spotlight support.

? Tracker integration must be setup and the share must be indexed by Tracker.

For a detailed set of instructions please see https://wiki.samba.org/index.php/Spotlight.

Default: spotlight = no

spotlight backend (S)

Spotlight search backend. Available backends:

? noindex - a backend that returns no results.

? tracker - Gnome Tracker.

? elasticsearch - a backend that uses JSON and REST over HTTP(s) to query an Elasticsearch server.

Default: spotlight backend = noindex

stat cache (G)

This parameter determines if smbd(8) will use a cache in order to speed up case insensitive name mappings. You should never need to change this parameter.

Default: stat cache = yes

state directory (G)

Usually, most of the TDB files are stored in the lock directory. Since Samba 3.4.0, it is possible to differentiate between TDB files with persistent data and TDB files with non-persistent data using the state directory and the cache directory options.

This option specifies the directory where TDB files containing important persistent data will be stored.

Default: state directory = /var/lib/samba

Example: state directory = /var/run/samba/locks/state

store dos attributes (S)

If this parameter is set Samba attempts to first read DOS attributes (SYSTEM, HIDDEN, ARCHIVE or READ-ONLY) from a filesystem extended attribute, before mapping DOS attributes to UNIX permission bits (such as occurs with map hidden and map readonly). When set, DOS attributes will be stored onto an extended attribute in the UNIX filesystem, associated with the file or directory. When this parameter is set it will override the parameters map hidden, map system, map archive and map readonly and they will behave as if they were set to off. This parameter writes the DOS attributes as a string into the extended attribute named "user.DOSATTRIB". This extended attribute is explicitly hidden from smbd clients requesting an EA list. On Linux the filesystem must have been mounted with the mount option user_xattr in order for extended attributes to work, also extended attributes must be compiled into the Linux kernel. In Samba 3.5.0 and above the "user.DOSATTRIB" extended attribute has been extended to store the create time for a file as well as the DOS attributes. This is done in a backwards compatible way so files created by Samba 3.5.0 and above can still have the DOS attribute read from this extended attribute by earlier versions of Samba, but they will not be able to read the create time stored there. Storing the create time separately from the normal filesystem meta-data allows Samba to faithfully reproduce NTFS semantics on top of a POSIX filesystem. The default has changed to yes in Samba release 4.9.0 and above to allow better Windows fileserver compatibility in a default install.

Default: store dos attributes = yes

strict allocate (S)

This is a boolean that controls the handling of disk space allocation in the server. When this is set to yes the server will change from UNIX behaviour of not committing real disk storage blocks when a file is extended to the Windows behaviour of actually forcing the disk system to allocate real storage blocks when a file is created or extended to be a given size. In UNIX terminology this means that Samba will stop creating sparse files.

This option is really designed for file systems that support fast allocation of large numbers of blocks such as extent-based file systems. On file systems that don't support extents (most notably ext3) this can make Samba slower. When you work with large files over >100MB on file systems without extents you may even run into problems with clients running into timeouts.

When you have an extent based filesystem it's likely that we can make use of unwritten extents which allows Samba to allocate even large amounts of space very fast and you will not see any timeout problems caused by strict allocate. With strict allocate in use you will also get much better out of quota messages in case you use quotas. Another advantage of activating this setting is that it will help to reduce file fragmentation.

To give you an idea on which filesystems this setting might currently be a good option for you: XFS, ext4, btrfs, ocfs2 on Linux and JFS2 on AIX support unwritten extents. On Filesystems that do not support it, preallocation is probably an expensive operation where you will see reduced performance and risk to let clients run into timeouts when creating large files. Examples are ext3, ZFS, HFS+ and most others, so be aware if you activate this setting on those filesystems.

Default: strict allocate = no

strict locking (S)

This is an enumerated type that controls the handling of file locking in the server. When this is set to yes, the server will check every read and write access for file locks, and deny access if locks exist. This can be slow on some systems.

When strict locking is set to Auto (the default), the server performs file lock checks only on non-oplocked files. As most Windows redirectors perform file locking checks locally on oplocked files this is a good trade off for improved performance.

When strict locking is disabled, the server performs file lock checks only when the client explicitly asks for them.

Well-behaved clients always ask for lock checks when it is important. So in the vast majority of cases, strict locking = Auto or strict locking = no is acceptable.

Default: strict locking = Auto

strict rename (S)

By default a Windows SMB server prevents directory renames when there are open file or directory handles below it in the filesystem hierarchy. Historically Samba has always allowed this as POSIX filesystem semantics require it.

This boolean parameter allows Samba to match the Windows behavior. Setting this to "yes" is a very expensive change, as it forces Samba to travers the entire open file handle database on every directory rename request. In a clustered Samba system the cost is even greater than the non-clustered case.

When set to "no" smbd only checks the local process the client is attached to for open files below a directory being renamed, instead of checking for open files across all smbd processes.

Because of the expense in fully searching the database, the default is "no", and it is recommended to be left that way unless a specific Windows application requires it to be changed.

If the client has requested UNIX extensions (POSIX pathnames) then renames are always allowed and this parameter has no effect.

Default: strict rename = no

strict sync (S)

This parameter controls whether Samba honors a request from an SMB client to ensure any outstanding operating system buffer contents held in memory are safely written onto stable storage on disk. If set to yes, which is the default, then Windows applications can force the smbd server to synchronize unwritten data onto the disk. If set to no then smbd will ignore client requests to synchronize unwritten data onto stable storage on disk.

In Samba 4.7.0, the default for this parameter changed from no to yes to better match the expectations of SMB2/3 clients and improve application safety when running against smbd.

The flush request from SMB2/3 clients is handled asynchronously inside smbd, so leaving the parameter as the default value of yes does not block the processing of other requests to the smbd process.

Legacy Windows applications (such as the Windows 98 explorer shell) seemed to confuse writing buffer contents to the operating system with synchronously writing outstanding data onto stable storage on disk. Changing this parameter to no means that smbd(8) will ignore the Windows applications request to synchronize unwritten data onto disk. Only consider changing this if smbd is serving obsolete SMB1 Windows clients prior to Windows XP (Windows 98 and below). There should be no need to change this setting for normal operations.

Default: strict sync = yes

svcctl list (G)

This option defines a list of init scripts that smbd will use for starting and stopping Unix services via the Win32 ServiceControl API. This allows Windows administrators to utilize the MS Management Console plug-ins to manage a Unix server running Samba.

The administrator must create a directory name svcctl in Samba's $(libdir) and create symbolic links to the init scripts in /etc/init.d/. The name of the links must match the names given as part of the svcctl list.

Default: svcctl list =

Example: svcctl list = cups postfix portmap httpd

sync always (S)

This is a boolean parameter that controls whether writes will always be written to stable storage before the write call returns. If this is no then the server will be guided by the client's request in each write call (clients can set a bit indicating that a particular write should be synchronous). If this is yes then every write will be followed by a fsync() call to ensure the data is written to disk. Note that the strict sync parameter must be set to yes in order for this parameter to have any effect.

Default: sync always = no

syslog (G)

This parameter maps how Samba debug messages are logged onto the system syslog logging levels. Samba debug level zero maps onto syslog LOG_ERR, debug level one maps onto LOG_WARNING, debug level two maps onto LOG_NOTICE, debug level three maps onto LOG_INFO. All higher levels are mapped to LOG_DEBUG.

This parameter sets the threshold for sending messages to syslog. Only messages with debug level less than this value will be sent to syslog. There still will be some logging to log.[sn]mbd even if syslog only is enabled.

The logging parameter should be used instead. When logging is set, it overrides the syslog parameter.

Default: syslog = 1

syslog only (G)

If this parameter is set then Samba debug messages are logged into the system syslog only, and not to the debug log files. There still will be some logging to log.[sn]mbd even if syslog only is enabled. The logging parameter should be used instead. When logging is set, it overrides the syslog only parameter.

Default: syslog only = no

template homedir (G)

When filling out the user information for a Windows NT user, the winbindd(8) daemon uses this parameter to fill in the home directory for that user. If the string %D is present it is substituted with the user's Windows NT domain name. If the string %U is present it is substituted with the user's Windows NT user name.

Default: template homedir = /home/%D/%U

template shell (G)

When filling out the user information for a Windows NT user, the winbindd(8) daemon uses this parameter to fill in the login shell for that user.

Default: template shell = /bin/false

time server (G)

This parameter determines if nmbd(8) advertises itself as a time server to Windows clients.

Default: time server = no

debug timestamp

This parameter is a synonym for timestamp logs.

timestamp logs (G)

Samba debug log messages are timestamped by default. If you are running at a high debug level these timestamps can be distracting. This boolean parameter allows timestamping to be turned off.

Default: timestamp logs = yes

tls cafile (G)

This option can be set to a file (PEM format) containing CA certificates of root CAs to trust to sign certificates or intermediate CA certificates.

This path is relative to private dir if the path does not start with a /.

Default: tls cafile = tls/ca.pem

tls certfile (G)

This option can be set to a file (PEM format) containing the RSA certificate.

This path is relative to private dir if the path does not start with a /.

Default: tls certfile = tls/cert.pem

tls crlfile (G)

This option can be set to a file containing a certificate revocation list (CRL).

This path is relative to private dir if the path does not start with a /.

Default: tls crlfile =

tls dh params file (G)

This option can be set to a file with Diffie-Hellman parameters which will be used with DH ciphers.

This path is relative to private dir if the path does not start with a /.

Default: tls dh params file =

tls enabled (G)

If this option is set to yes, then Samba will use TLS when possible in communication.

Default: tls enabled = yes

tls keyfile (G)

This option can be set to a file (PEM format) containing the RSA private key. This file must be accessible without a pass-phrase, i.e. it must not be encrypted.

This path is relative to private dir if the path does not start with a /.

Default: tls keyfile = tls/key.pem

tls priority (G)

This option can be set to a string describing the TLS protocols to be supported in the parts of Samba that use GnuTLS, specifically the AD DC.

The string is appended to the default priority list of GnuTLS.

The valid options are described in the GNUTLS Priority-Strings documentation at

http://gnutls.org/manual/html_node/Priority-Strings.html

The SSL3.0 protocol will be disabled.

Default: tls priority = NORMAL:-VERS-SSL3.0

tls verify peer (G)

This controls if and how strict the client will verify the peer's certificate and name. Possible values are (in increasing order): no_check, ca_only, ca_and_name_if_available, ca_and_name and as_strict_as_possible.

When set to no_check the certificate is not verified at all, which allows trivial man in the middle attacks.

When set to ca_only the certificate is verified to be signed from a ca specified in the tls ca file option. Setting tls ca file to a valid file is required. The certificate lifetime is also verified. If the tls crl file option is configured, the certificate is also verified against the ca crl.

When set to ca_and_name_if_available all checks from ca_only are performed. In addition, the peer hostname is verified against the certificate's name, if it is provided by the application layer and not given as an ip address string.

When set to ca_and_name all checks from ca_and_name_if_available are performed. In addition the peer hostname needs to be provided and even an ip address is checked against the certificate's name.

When set to as_strict_as_possible all checks from ca_and_name are performed. In addition the tls crl file needs to be configured.

Future versions of Samba may implement additional checks.

Default: tls verify peer = as_strict_as_possible

unicode (G)

Specifies whether the server and client should support unicode.

If this option is set to false, the use of ASCII will be forced.

Default: unicode = yes

unix charset (G)

Specifies the charset the unix machine Samba runs on uses. Samba needs to know this in order to be able to convert text to the charsets other SMB clients use.

This is also the charset Samba will use when specifying arguments to scripts that it invokes.

Default: unix charset = UTF-8

Example: unix charset = ASCII

unix password sync (G)

This boolean parameter controls whether Samba attempts to synchronize the UNIX password with the SMB password when the encrypted SMB password in the smbpasswd file is changed. If this is set to yes the program specified in the passwd program parameter is called AS ROOT - to allow the new UNIX password to be set without access to the old UNIX password (as the SMB password change code has no access to the old password cleartext, only the new).

This option has no effect if samba is running as an active directory domain controller, in that case have a look at the password hash gpg key ids option and the samba-tool user syncpasswords command.

Default: unix password sync = no

use client driver (S)

This parameter applies only to Windows NT/2000 clients. It has no effect on Windows 95/98/ME clients. When serving a printer to Windows NT/2000 clients without first installing a valid printer driver on the Samba host, the client will be required to install a local printer driver. From this point on, the client will treat the print as a local printer and not a network printer connection. This is much the same behavior that will occur when disable spoolss = yes.

The differentiating factor is that under normal circumstances, the NT/2000 client will attempt to open the network printer using MS-RPC. The problem is that because the client considers the printer to be local, it will attempt to issue the OpenPrinterEx() call requesting access rights associated with the logged on user. If the user possesses local administrator rights but not root privilege on the Samba host (often the case), the OpenPrinterEx() call will fail. The result is that the client will now display an "Access Denied; Unable to connect" message in the printer queue window (even though jobs may successfully be printed).

If this parameter is enabled for a printer, then any attempt to open the printer with the PRINTER_ACCESS_ADMINISTER right is mapped to PRINTER_ACCESS_USE instead. Thus allowing the OpenPrinterEx() call to succeed. This parameter MUST not be enabled on a print share which has valid print driver installed on the Samba server.

Default: use client driver = no

use mmap (G)

This global parameter determines if the tdb internals of Samba can depend on mmap working correctly on the running system. Samba requires a coherent mmap/read-write system memory cache. Currently only OpenBSD and HPUX do not have such a coherent cache, and on those platforms this parameter is overridden internally to be effeceively no. On all systems this parameter should be left alone. This parameter is provided to help the Samba developers track down problems with the tdb internal code.

Default: use mmap = yes

username level (G)

This option helps Samba to try and 'guess' at the real UNIX username, as many DOS clients send an all-uppercase username. By default Samba tries all lowercase, followed by the username with the first letter capitalized, and fails if the username is not found on the UNIX machine.

If this parameter is set to non-zero the behavior changes. This parameter is a number that specifies the number of uppercase combinations to try while trying to determine the UNIX user name. The higher the number the more combinations will be tried, but the slower the discovery of usernames will be. Use this parameter when you have strange usernames on your UNIX machine, such as AstrangeUser .

This parameter is needed only on UNIX systems that have case sensitive usernames.

Default: username level = 0

Example: username level = 5

username map (G)

This option allows you to specify a file containing a mapping of usernames from the clients to the server. This can be used for several purposes. The most common is to map usernames that users use on DOS or Windows machines to those that the UNIX box uses. The other is to map multiple users to a single username so that they can more easily share files.

Please note that for user mode security, the username map is applied prior to validating the user credentials. Domain member servers (domain or ads) apply the username map after the user has been successfully authenticated by the domain controller and require fully qualified entries in the map table (e.g. biddle = DOMAIN\foo).

The map file is parsed line by line. Each line should contain a single UNIX username on the left then a '=' followed by a list of usernames on the right. The list of usernames on the right may contain names of the form @group in which case they will match any UNIX username in that group. The special client name '*' is a wildcard and matches any name. Each line of the map file may be up to 1023 characters long.

The file is processed on each line by taking the supplied username

and comparing it with each username on the right hand side of the

'=' signs. If the supplied name matches any of the names on the

right hand side then it is replaced with the name on the left.

Processing then continues with the next line.

If any line begins with a '#' or a ';' then it is ignored.

If any line begins with an '!' then the processing will stop after

that line if a mapping was done by the line. Otherwise mapping

continues with every line being processed. Using '!' is most useful

when you have a wildcard mapping line later in the file.

For example to map from the name admin or administrator to the UNIX

name

root you would use:

```
root = admin administrator
```

Or to map anyone in the UNIX group system to the UNIX name sys you

would use:

```
sys = @system
```

You can have as many mappings as you like in a username map file.

If your system supports the NIS NETGROUP option then the netgroup

database is checked before the /etc/group database for matching

groups.

You can map Windows usernames that have spaces in them by using

double quotes around the name. For example:

```
tridge = "Andrew Tridgell"
```

would map the windows username "Andrew Tridgell" to the unix

username "tridge".

The following example would map mary and fred to the unix user sys,

and map the rest to guest. Note the use of the '!' to tell Samba to

stop processing if it gets a match on that line:

```
!sys = mary fred
guest = *
```

Note that the remapping is applied to all occurrences of usernames.

Thus if you connect to \\server\fred and fred is remapped to mary then you will actually be connecting to \\server\mary and will need to supply a password suitable for mary not fred. The only exception to this is the username passed to a Domain Controller (if you have one). The DC will receive whatever username the client supplies without modification.

Also note that no reverse mapping is done. The main effect this has is with printing. Users who have been mapped may have trouble deleting print jobs as PrintManager under WfWg will think they don't own the print job.

Samba versions prior to 3.0.8 would only support reading the fully qualified username (e.g.: DOMAIN\user) from the username map when performing a kerberos login from a client. However, when looking up a map entry for a user authenticated by NTLM[SSP], only the login name would be used for matches. This resulted in inconsistent behavior sometimes even on the same server.

The following functionality is obeyed in version 3.0.8 and later:

When performing local authentication, the username map is applied to the login name before attempting to authenticate the connection.

When relying upon a external domain controller for validating authentication requests, smbd will apply the username map to the fully qualified username (i.e. DOMAIN\user) only after the user has been successfully authenticated.

An example of use is:

```
    username map = /usr/local/samba/lib/users.map
```

Default: username map = # no username map

username map cache time (G)

Mapping usernames with the username map or username map script features of Samba can be relatively expensive. During login of a user, the mapping is done several times. In particular, calling the username map script can slow down logins if external databases have to be queried from the script being called.

The parameter username map cache time controls a mapping cache. It specifies the number of seconds a mapping from the username map file or script is to be efficiently cached. The default of 0 means no caching is done.

Default: username map cache time = 0

Example: username map cache time = 60

username map script (G)

This script is a mutually exclusive alternative to the username map parameter. This parameter specifies an external program or script that must accept a single command line option (the username transmitted in the authentication request) and return a line on standard output (the name to which the account should mapped). In this way, it is possible to store username map tables in an LDAP directory services.

Default: username map script =

Example: username map script = /etc/samba/scripts/mapusers.sh

usershare allow guests (G)

This parameter controls whether user defined shares are allowed to be accessed by non-authenticated users or not. It is the equivalent of allowing people who can create a share the option of setting guest ok = yes in a share definition. Due to its security sensitive nature, the default is set to off.

Default: usershare allow guests = no

usershare max shares (G)

This parameter specifies the number of user defined shares that are allowed to be created by users belonging to the group owning the usershare directory. If set to zero (the default) user defined shares are ignored.

Default: usershare max shares = 0

usershare owner only (G)

This parameter controls whether the pathname exported by a user defined shares must be owned by the user creating the user defined share or not. If set to True (the default) then smbd checks that the directory path being shared is owned by the user who owns the usershare file defining this share and refuses to create the share if not. If set to False then no such check is performed and any directory path may be exported regardless of who owns it.

Default: usershare owner only = yes

usershare path (G)

This parameter specifies the absolute path of the directory on the filesystem used to store the user defined share definition files.

This directory must be owned by root, and have no access for other, and be writable only by the group owner. In addition the "sticky" bit must also be set, restricting rename and delete to owners of a file (in the same way the /tmp directory is usually configured).

Members of the group owner of this directory are the users allowed to create usershares.

For example, a valid usershare directory might be /usr/local/samba/lib/usershares, set up as follows.

```
ls -ld /usr/local/samba/lib/usershares/
drwxrwx--T  2 root power_users 4096 2006-05-05 12:27 /usr/local/samba/lib/usershares/
```

In this case, only members of the group "power_users" can create user defined shares.

Default: usershare path = /var/lib/samba/usershares

usershare prefix allow list (G)

This parameter specifies a list of absolute pathnames the root of which are allowed to be exported by user defined share definitions.

If the pathname to be exported doesn't start with one of the strings in this list, the user defined share will not be allowed.

This allows the Samba administrator to restrict the directories on the system that can be exported by user defined shares.

If there is a "usershare prefix deny list" and also a "usershare prefix allow list" the deny list is processed first, followed by the allow list, thus leading to the most restrictive interpretation.

Default: usershare prefix allow list =

Example: usershare prefix allow list = /home /data /space

usershare prefix deny list (G)

This parameter specifies a list of absolute pathnames the root of which are NOT allowed to be exported by user defined share definitions. If the pathname exported starts with one of the strings in this list the user defined share will not be allowed. Any pathname not starting with one of these strings will be allowed to be exported as a usershare. This allows the Samba administrator to restrict the directories on the system that can be exported by user defined shares.

If there is a "usershare prefix deny list" and also a "usershare prefix allow list" the deny list is processed first, followed by the allow list, thus leading to the most restrictive interpretation.

Default: usershare prefix deny list =

Example: usershare prefix deny list = /etc /dev /private

usershare template share (G)

User defined shares only have limited possible parameters such as path, guest ok, etc. This parameter allows usershares to "cloned" from an existing share. If "usershare template share" is set to the name of an existing share, then all usershares created have their defaults set from the parameters set on this share.

The target share may be set to be invalid for real file sharing by setting the parameter "-valid = False" on the template share definition. This causes it not to be seen as a real exported share but to be able to be used as a template for usershares.

Default: usershare template share =

Example: usershare template share = template_share

use sendfile (S)

If this parameter is yes, and the sendfile() system call is supported by the underlying operating system, then some SMB read calls (mainly ReadAndX and ReadRaw) will use the more efficient sendfile system call for files that are exclusively oplocked. This may make more efficient use of the system CPU's and cause Samba to be faster. Samba automatically turns this off for clients that use protocol levels lower than NT LM 0.12 and when it detects a client is Windows 9x (using sendfile from Linux will cause these clients to fail).

Default: use sendfile = no

utmp (G)

This boolean parameter is only available if Samba has been configured and compiled with the option --with-utmp. If set to yes then Samba will attempt to add utmp or utmpx records (depending on the UNIX system) whenever a connection is made to a Samba server. Sites may use this to record the user connecting to a Samba share.

Due to the requirements of the utmp record, we are required to create a unique identifier for the incoming user. Enabling this option creates an n^2 algorithm to find this number. This may impede performance on large installations.

Default: utmp = no

utmp directory (G)

This parameter is only available if Samba has been configured and compiled with the option --with-utmp. It specifies a directory pathname that is used to store the utmp or utmpx files (depending on the UNIX system) that record user connections to a Samba server. By default this is not set, meaning the system will use whatever utmp file the native system is set to use (usually /var/run/utmp on Linux).

Default: utmp directory = # Determined automatically

Example: utmp directory = /var/run/utmp

-valid (S)

This parameter indicates whether a share is valid and thus can be used. When this parameter is set to false, the share will be in no way visible nor accessible.

This option should not be used by regular users but might be of help to developers. Samba uses this option internally to mark shares as deleted.

Default: -valid = yes

valid users (S)

This is a list of users that should be allowed to login to this service. Names starting with '@', '+' and '&' are interpreted using the same rules as described in the invalid users parameter.

If this is empty (the default) then any user can login. If a username is in both this list and the invalid users list then access is denied for that user.

The current servicename is substituted for %S. This is useful in the [homes] section.

Note: When used in the [global] section this parameter may have unwanted side effects. For example: If samba is configured as a MASTER BROWSER (see local master, os level, domain master, preferred master) this option will prevent workstations from being able to browse the network.

Default: valid users = # No valid users list (anyone can login)

Example: valid users = greg, @pcusers

veto files (S)

This is a list of files and directories that are neither visible nor accessible. Each entry in the list must be separated by a '/', which allows spaces to be included in the entry. '*' and '?' can be used to specify multiple files or directories as in DOS wildcards.

Each entry must be a unix path, not a DOS path and must not include the unix directory separator '/'.

Note that the case sensitive option is applicable in vetoing files.

One feature of the veto files parameter that it is important to be aware of is Samba's behaviour when trying to delete a directory. If a directory that is to be deleted contains nothing but veto files this deletion will fail unless you also set the delete veto files parameter to yes.

Setting this parameter will affect the performance of Samba, as it will be forced to check all files and directories for a match as they are scanned.

Examples of use include:

```
; Veto any files containing the word Security,
; any ending in .tmp, and any directory containing the
; word root.
veto files = /*Security*/*.tmp/*root*/
; Veto the Apple specific files that a NetAtalk server
; creates.
veto files = /.AppleDouble/.bin/.AppleDesktop/Network Trash Folder/
```

Default: veto files = # No files or directories are vetoed

veto oplock files (S)

This parameter is only valid when the oplocks parameter is turned on for a share. It allows the Samba administrator to selectively turn off the granting of oplocks on selected files that match a wildcarded list, similar to the wildcarded list used in the veto files parameter.

You might want to do this on files that you know will be heavily contended for by clients. A good example of this is in the NetBench SMB benchmark program, which causes heavy client contention for files ending in .SEM. To cause Samba not to grant oplocks on these files you would use the line (either in the [global] section or in the section for the particular NetBench share.

An example of use is:

```
veto oplock files = /.*SEM/
```

Default: veto oplock files = # No files are vetoed for oplock grants

vfs object

This parameter is a synonym for vfs objects.

vfs objects (S)

This parameter specifies the backend names which are used for Samba VFS I/O operations. By default, normal disk I/O operations are used but these can be overloaded with one or more VFS objects. Be aware that the definition of this parameter will overwrite a possible previous definition of the vfs objects parameter.

Default: vfs objects =

Example: vfs objects = extd_audit recycle

volume (S)

This allows you to override the volume label returned for a share.

Useful for CDROMs with installation programs that insist on a

particular volume label.

Default: volume = # the name of the share

volume serial number (S)

This allows to override the volume serial number (a 32bit value)

reported for a share.

The special value -1 (default) stands for a unique number that is

calculated for each share.

Default: volume serial number = -1

Example: volume serial number = 0xabcdefgh

wide links (S)

This parameter controls whether or not links in the UNIX file

system may be followed by the server. Links that point to areas

within the directory tree exported by the server are always

allowed; this parameter controls access only to areas that are

outside the directory tree being exported.

Note: Turning this parameter on when UNIX extensions are enabled

will allow UNIX clients to create symbolic links on the share that

can point to files or directories outside restricted path exported

by the share definition. This can cause access to areas outside of

the share. Due to this problem, this parameter will be

automatically disabled (with a message in the log file) if the unix

extensions option is on.

See the parameter allow insecure wide links if you wish to change

this coupling between the two parameters.

Default: wide links = no

winbind cache time (G)

This parameter specifies the number of seconds the winbindd(8)

daemon will cache user and group information before querying a

Windows NT server again.

This does not apply to authentication requests, these are always evaluated in real time unless the winbind offline logon option has been enabled.

Default: winbind cache time = 300

winbindd socket directory (G)

This setting controls the location of the winbind daemon's socket. Except within automated test scripts, this should not be altered, as the client tools (nss_winbind etc) do not honour this parameter. Client tools must then be advised of the altered path with the WINBINDD_SOCKET_DIR environment variable.

Default: winbindd socket directory = /run/samba/winbindd

winbind enum groups (G)

On large installations using winbindd(8) it may be necessary to suppress the enumeration of groups through the setgrent(), getgrent() and endgrent() group of system calls. If the winbind enum groups parameter is no, calls to the getgrent() system call will not return any data.

Warning

Turning off group enumeration may cause some programs to behave oddly.

Default: winbind enum groups = no

winbind enum users (G)

On large installations using winbindd(8) it may be necessary to suppress the enumeration of users through the setpwent(), getpwent() and endpwent() group of system calls. If the winbind enum users parameter is no, calls to the getpwent system call will not return any data.

Warning

Turning off user enumeration may cause some programs to behave oddly. For example, the finger program relies on having access to the full user list when searching for matching usernames.

Default: winbind enum users = no

winbind expand groups (G)

This option controls the maximum depth that winbindd will traverse when flattening nested group memberships of Windows domain groups. This is different from the winbind nested groups option which implements the Windows NT4 model of local group nesting. The "winbind expand groups" parameter specifically applies to the membership of domain groups.

This option also affects the return of non nested group memberships of Windows domain users. With the new default "winbind expand groups = 0" winbind does not query group memberships at all.

Be aware that a high value for this parameter can result in system slowdown as the main parent winbindd daemon must perform the group unrolling and will be unable to answer incoming NSS or authentication requests during this time.

The default value was changed from 1 to 0 with Samba 4.2. Some broken applications (including some implementations of newgrp and sg) calculate the group memberships of users by traversing groups, such applications will require "winbind expand groups = 1". But the new default makes winbindd more reliable as it doesn't require SAMR access to domain controllers of trusted domains.

Default: winbind expand groups = 0

winbind:ignore domains (G)

Allows one to enter a list of trusted domains winbind should ignore (untrust). This can avoid the overhead of resources from attempting to login to DCs that should not be communicated with.

Default: winbind:ignore domains =

Example: winbind:ignore domains = DOMAIN1, DOMAIN2

winbind max clients (G)

This parameter specifies the maximum number of clients the winbindd(8) daemon can connect with. The parameter is not a hard limit. The winbindd(8) daemon configures itself to be able to accept at least that many connections, and if the limit is reached, an attempt is made to disconnect idle clients.

Default: winbind max clients = 200

winbind max domain connections (G)

This parameter specifies the maximum number of simultaneous connections that the winbindd(8) daemon should open to the domain controller of one domain. Setting this parameter to a value greater than 1 can improve scalability with many simultaneous winbind requests, some of which might be slow.

Note that if winbind offline logon is set to Yes, then only one DC connection is allowed per domain, regardless of this setting.

Default: winbind max domain connections = 1

Example: winbind max domain connections = 10

winbind nested groups (G)

If set to yes, this parameter activates the support for nested groups. Nested groups are also called local groups or aliases. They work like their counterparts in Windows: Nested groups are defined locally on any machine (they are shared between DC's through their SAM) and can contain users and global groups from any trusted SAM. To be able to use nested groups, you need to run nss_winbind.

Default: winbind nested groups = yes

winbind normalize names (G)

This parameter controls whether winbindd will replace whitespace in user and group names with an underscore (_) character. For example, whether the name "Space Kadet" should be replaced with the string "space_kadet". Frequently Unix shell scripts will have difficulty with usernames contains whitespace due to the default field separator in the shell. If your domain possesses names containing the underscore character, this option may cause problems unless the name aliasing feature is supported by your nss_info plugin.

This feature also enables the name aliasing API which can be used to make domain user and group names to a non-qualified version. Please refer to the manpage for the configured idmap and nss_info plugin for the specifics on how to configure name aliasing for a specific configuration. Name aliasing takes precedence (and is mutually exclusive) over the whitespace replacement mechanism discussed previously.

Default: winbind normalize names = no

Example: winbind normalize names = yes

winbind nss info (G)

This parameter is designed to control how Winbind retrieves Name Service Information to construct a user's home directory and login shell. Currently the following settings are available:

? template - The default, using the parameters of template shell and template homedir)

? <sfu | sfu20 | rfc2307 > - When Samba is running in security = ads and your Active Directory Domain Controller does support the Microsoft "Services for Unix" (SFU) LDAP schema, winbind can retrieve the login shell and the home directory attributes directly from your Directory Server. For SFU 3.0 or 3.5 simply choose "sfu", if you use SFU 2.0 please choose "sfu20". Note that for the idmap backend idmap_ad you need to configure those settings in the idmap configuration section. Make sure to consult the documentation of the idmap backend that you are using.

Default: winbind nss info = template

Example: winbind nss info = sfu

winbind offline logon (G)

This parameter is designed to control whether Winbind should allow one to login with the pam_winbind module using Cached Credentials. If enabled, winbindd will store user credentials from successful logins encrypted in a local cache.

Default: winbind offline logon = no

Example: winbind offline logon = yes

winbind reconnect delay (G)

This parameter specifies the number of seconds the winbindd(8) daemon will wait between attempts to contact a Domain controller for a domain that is determined to be down or not contactable.

Default: winbind reconnect delay = 30

winbind refresh tickets (G)

This parameter is designed to control whether Winbind should refresh Kerberos Tickets retrieved using the pam_winbind module.

Default: winbind refresh tickets = no

Example: winbind refresh tickets = yes

winbind request timeout (G)

This parameter specifies the number of seconds the winbindd(8) daemon will wait before disconnecting either a client connection with no outstanding requests (idle) or a client connection with a request that has remained outstanding (hung) for longer than this number of seconds.

Default: winbind request timeout = 60

winbind rpc only (G)

Setting this parameter to yes forces winbindd to use RPC instead of LDAP to retrieve information from Domain Controllers.

Default: winbind rpc only = no

winbind scan trusted domains (G)

This option only takes effect when the security option is set to domain or ads. If it is set to yes, winbindd periodically tries to scan for new trusted domains and adds them to a global list inside of winbindd. The list can be extracted with wbinfo --trusted-domains --verbose. Setting it to yes matches the behaviour of Samba 4.7 and older.

The construction of that global list is not reliable and often incomplete in complex trust setups. In most situations the list is not needed any more for winbindd to operate correctly. E.g. for plain file serving via SMB using a simple idmap setup with autorid, tdb or ad. However some more complex setups require the list, e.g. if you specify idmap backends for specific domains. Some pam_winbind setups may also require the global list.

If you have a setup that doesn't require the global list, you should set winbind scan trusted domains = no.

Default: winbind scan trusted domains = no

winbind sealed pipes (G)

This option controls whether any requests from winbindd to domain controllers pipe will be sealed. Disabling sealing can be useful for debugging purposes.

The behavior can be controlled per netbios domain by using 'winbind sealed pipes:NETBIOSDOMAIN = no' as option.

Default: winbind sealed pipes = yes

winbind separator (G)

This parameter allows an admin to define the character used when listing a username of the form of DOMAIN \user. This parameter is only applicable when using the pam_winbind.so and nss_winbind.so modules for UNIX services.

Please note that setting this parameter to + causes problems with group membership at least on glibc systems, as the character + is used as a special character for NIS in /etc/group.

Default: winbind separator = \

Example: winbind separator = +

winbind use default domain (G)

This parameter specifies whether the winbindd(8) daemon should operate on users without domain component in their username. Users without a domain component are treated as is part of the winbindd server's own domain. While this does not benefit Windows users, it makes SSH, FTP and e-mail function in a way much closer to the way they would in a native unix system.

This option should be avoided if possible. It can cause confusion about responsibilities for a user or group. In many situations it is not clear whether winbind or /etc/passwd should be seen as authoritative for a user, likewise for groups.

Default: winbind use default domain = no

Example: winbind use default domain = yes

winbind use krb5 enterprise principals (G)

winbindd is able to get kerberos tickets for pam_winbind with krb5_auth or wbinfo -K/--krb5auth=.

winbindd (at least on a domain member) is never be able to have a complete picture of the trust topology (which is managed by the DCs). There might be uPNSuffixes and msDS-SPNSuffixes values, which don't belong to any AD domain at all.

With winbind scan trusted domains = no winbindd doesn't even get a complete picture of the topology.

It is not really required to know about the trust topology. We can just rely on the [K]DCs of our primary domain (e.g. PRIMARY.A.EXAMPLE.COM) and use enterprise principals e.g. upnfromB@B.EXAMPLE.COM@PRIMARY.A.EXAMPLE.COM and follow the WRONG_REALM referrals in order to find the correct DC. The final principal might be userfromB@INTERNALB.EXAMPLE.PRIVATE.

With winbind use krb5 enterprise principals = yes winbindd enterprise principals will be used.

Default: winbind use krb5 enterprise principals = yes

Example: winbind use krb5 enterprise principals = no

winsdb:local_owner (G)

This specifies the address that is stored in the winsOwner attribute, of locally registered winsRecord-objects. The default is to use the ip-address of the first network interface.

No default

winsdb:dbnosync (G)

This parameter disables fsync() after changes of the WINS database.

Default: winsdb:dbnosync = no

wins hook (G)

When Samba is running as a WINS server this allows you to call an external program for all changes to the WINS database. The primary use for this option is to allow the dynamic update of external name resolution databases such as dynamic DNS.

The wins hook parameter specifies the name of a script or executable that will be called as follows:

wins_hook operation name nametype ttl IP_list

? The first argument is the operation and is one of "add", "delete", or "refresh". In most cases the operation can be ignored as the rest of the parameters provide sufficient information. Note that "refresh" may sometimes be called when the name has not previously been added, in that case it should be treated as an add.

? The second argument is the NetBIOS name. If the name is not a legal name then the wins hook is not called. Legal names contain only letters, digits, hyphens, underscores and periods.

? The third argument is the NetBIOS name type as a 2 digit hexadecimal number.

? The fourth argument is the TTL (time to live) for the name in seconds.

? The fifth and subsequent arguments are the IP addresses currently registered for that name. If this list is empty then the name should be deleted.

An example script that calls the BIND dynamic DNS update program nsupdate is provided in the examples directory of the Samba source code.

No default

wins proxy (G)

This is a boolean that controls if nmbd(8) will respond to broadcast name queries on behalf of other hosts. You may need to set this to yes for some older clients.

Default: wins proxy = no

wins server (G)

This specifies the IP address (or DNS name: IP address for preference) of the WINS server that nmbd(8) should register with. If you have a WINS server on your network then you should set this to the WINS server's IP.

You should point this at your WINS server if you have a multi-subnetted network.

If you want to work in multiple namespaces, you can give every wins server a 'tag'. For each tag, only one (working) server will be queried for a name. The tag should be separated from the ip address by a colon.

> Note
>
> You need to set up Samba to point to a WINS server if you have multiple subnets and wish cross-subnet browsing to work correctly.

See the chapter in the Samba3-HOWTO on Network Browsing.

Default: wins server =

Example: wins server = mary:192.9.200.1 fred:192.168.3.199 mary:192.168.2.61 # For this example when querying a certain name, 192.19.200.1 will be asked first and if that doesn't respond 192.168.2.61. If either of those doesn't know the name 192.168.3.199 will be queried.

Example: wins server = 192.9.200.1 192.168.2.61

wins support (G)

This boolean controls if the nmbd(8) process in Samba will act as a WINS server. You should not set this to yes unless you have a multi-subnetted network and you wish a particular nmbd to be your WINS server. Note that you should NEVER set this to yes on more than one machine in your network.

Default: wins support = no

workgroup (G)

This controls what workgroup your server will appear to be in when queried by clients. Note that this parameter also controls the Domain name used with the security = domain setting.

Default: workgroup = WORKGROUP

Example: workgroup = MYGROUP

wreplsrv:periodic_interval (G)

This maximum interval in seconds between 2 periodically scheduled runs where we check for wins.ldb changes and do push notifications to our push partners. Also wins_config.ldb changes are checked in that interval and partner configuration reloads are done.

Default: wreplsrv:periodic_interval = 15

wreplsrv:propagate name releases (G)

If this parameter is enabled, then explicit (from the client) and implicit (via the scavenging) name releases are propagated to the other servers directly, even if there are still other addresses active, this applies to SPECIAL GROUP (2) and MULTIHOMED (3) entries. Also the replication conflict merge algorithm for SPECIAL GROUP (2) entries discards replica addresses where the address owner is the local server, if the address was not stored locally before. The merge result is propagated directly in case an address was discarded. A Windows servers doesn't propagate name releases of SPECIAL GROUP (2) and MULTIHOMED (3) entries directly, which means that Windows servers may return different results to name queries for SPECIAL GROUP (2) and MULTIHOMED (3) names. The option doesn't have much negative impact if Windows servers are around, but be aware that they might return unexpected results.

Default: wreplsrv:propagate name releases = no

wreplsrv:scavenging_interval (G)

This is the interval in s between 2 scavenging runs which clean up the WINS database and changes the states of expired name records. Defaults to half of the value of wreplsrv:renew_interval.

No default

wreplsrv:tombstone_extra_timeout (G)

This is the time in s the server needs to be up till we'll remove tombstone records from our database. Defaults to 3 days.

Default: wreplsrv:tombstone_extra_timeout = 259200

wreplsrv:tombstone_interval (G)

This is the interval in s till released records of the WINS server become tombstone. Defaults to 6 days.

Default: wreplsrv:tombstone_interval = 518400

wreplsrv:tombstone_timeout (G)

This is the interval in s till tombstone records are deleted from the WINS database. Defaults to 1 day.

Default: wreplsrv:tombstone_timeout = 86400

wreplsrv:verify_interval (G)

This is the interval in s till we verify active replica records

with the owning WINS server. Unfortunately not implemented yet.

Defaults to 24 days.

Default: wreplsrv:verify_interval = 2073600

writable

This parameter is a synonym for writeable.

write ok

This parameter is a synonym for writeable.

writeable (S)

Inverted synonym for read only.

Default: writeable = no

write list (S)

This is a list of users that are given read-write access to a

service. If the connecting user is in this list then they will be

given write access, no matter what the read only option is set to.

The list can include group names using the @group syntax.

Note that if a user is in both the read list and the write list

then they will be given write access.

Default: write list =

Example: write list = admin, root, @staff

write raw (G)

This is ignored if async smb echo handler is set, because this

feature is incompatible with raw write SMB requests

If enabled, raw writes allow writes of 65535 bytes in one packet.

This typically provides a major performance benefit for some very,

very old clients.

However, some clients either negotiate the allowable block size

incorrectly or are incapable of supporting larger block sizes, and

for these clients you may need to disable raw writes.

In general this parameter should be viewed as a system tuning tool and left severely alone.

Default: write raw = yes

wtmp directory (G)

This parameter is only available if Samba has been configured and compiled with the option --with-utmp. It specifies a directory pathname that is used to store the wtmp or wtmpx files (depending on the UNIX system) that record user connections to a Samba server. The difference with the utmp directory is the fact that user info is kept after a user has logged out.

By default this is not set, meaning the system will use whatever utmp file the native system is set to use (usually /var/run/wtmp on Linux).

Default: wtmp directory =

Example: wtmp directory = /var/log/wtmp

WARNINGS

Although the configuration file permits service names to contain spaces, your client software may not. Spaces will be ignored in comparisons anyway, so it shouldn't be a problem - but be aware of the possibility.

On a similar note, many clients - especially DOS clients - limit service names to eight characters. smbd(8) has no such limitation, but attempts to connect from such clients will fail if they truncate the service names. For this reason you should probably keep your service names down to eight characters in length.

Use of the [homes] and [printers] special sections make life for an administrator easy, but the various combinations of default attributes can be tricky. Take extreme care when designing these sections. In particular, ensure that the permissions on spool directories are correct.

VERSION

This man page is part of version 4.17.5 of the Samba suite.

SEE ALSO

samba(7), smbpasswd(8), smbd(8), nmbd(8), winbindd(8), samba(8), samba-tool(8), smbclient(1), nmblookup(1), testparm(1).

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.

Samba 4.17.5 07/18/2023 SMB.CONF(5)