Chapter 10

ACCOUNT INFORMATION DATABASES

Samba-3 implements a new capability to work concurrently with multiple account backends. The possible new combinations of password backends allows Samba-3 a degree of flexibility and scalability that previously could be achieved only with MS Windows Active Directory (ADS). This chapter describes the new functionality and how to get the most out of it.

The three passdb backends that are fully maintained (actively supported) by the Samba Team are: `smbpasswd` (being obsoleted), `tdbsam` (a tdb-based binary file format), and `ldapsam` (LDAP directory). Of these, only the `ldapsam` backend stores both POSIX (UNIX) and Samba user and group account information in a single repository. The `smbpasswd` and `tdbsam` backends store only Samba user accounts.

In a strict sense, there are three supported account storage and access systems. One of these is considered obsolete (smbpasswd). It is recommended to use the `tdbsam` method for all simple systems. Use `ldapsam` for larger and more complex networks.

In a strict and literal sense, the passdb backends are account storage mechanisms (or methods) alone. The choice of terminology can be misleading, however we are stuck with this choice of wording. This chapter documents the nature of the account storage system with a focus on user and trust accounts. Trust accounts have two forms, machine trust accounts (computer accounts) and interdomain trust accounts. These are all treated as user-like entities.

10.1 Features and Benefits

Samba-3 provides for complete backward compatibility with Samba-2.2.x functionality as follows:

10.1.1 Backward Compatibility Account Storage Systems

Plaintext This isn't really a backend at all, but is listed here for simplicity. Samba can be configured to pass plaintext authentication requests to the traditional UNIX/Linux `/etc/passwd` and `/etc/shadow`-style subsystems. On systems that have Pluggable Authentication Modules (PAM) support, all PAM modules are supported. The behavior is just as it was with Samba-2.2.x, and the protocol limitations imposed by MS Windows clients apply likewise. Please refer to Section 10.2, for more information regarding the limitations of plaintext password usage.

smbpasswd This option allows continued use of the `smbpasswd` file that maintains a plain ASCII (text) layout that includes the MS Windows LanMan and NT-encrypted passwords as well as a field that stores some account information. This form of password backend does not store any of the MS Windows NT/200x SAM (Security Account Manager) information required to provide the extended controls that are needed for more comprehensive interoperation with MS Windows NT4/200x servers.

This backend should be used only for backward compatibility with older versions of Samba. It may be deprecated in future releases.

ldapsam_compat (Samba-2.2 LDAP Compatibility) There is a password backend option that allows continued operation with an existing OpenLDAP backend that uses the Samba-2.2.x LDAP schema extension. This option is provided primarily as a migration tool, although there is no reason to force migration at this time. This tool will eventually be deprecated.

10.1.2 New Account Storage Systems

Samba-3 introduces a number of new password backend capabilities.

tdbsam This backend provides a rich database backend for local servers. This backend is not suitable for multiple domain controllers (i.e., PDC + one or more BDC) installations.

The *tdbsam* password backend stores the old *smbpasswd* information plus the extended MS Windows NT/200x SAM information into a binary format TDB (trivial database) file. The inclusion of the extended information makes it possible for Samba-3 to implement the same account and system access controls that are possible with MS Windows NT4/200x-based systems.

The inclusion of the *tdbsam* capability is a direct response to user requests to allow simple site operation without the overhead of the complexities of running OpenLDAP. It is recommended to use this only for sites that have fewer than 250 users. For larger sites or implementations, the use of OpenLDAP or of Active Directory integration is strongly recommended.

ldapsam This provides a rich directory backend for distributed account installation.

Samba-3 has a new and extended LDAP implementation that requires configuration of OpenLDAP with a new format Samba schema. The new format schema file is included in the `examples/LDAP` directory of the Samba distribution.

The new LDAP implementation significantly expands the control abilities that were possible with prior versions of Samba. It is now possible to specify "per-user" profile settings, home directories, account access controls, and much more. Corporate sites will see that the Samba Team has listened to their requests both for capability and greater scalability.

mysqlsam (MySQL-based backend) It is expected that the MySQL-based SAM will be very popular in some corners. This database backend will be of considerable interest to sites that want to leverage existing MySQL technology.

pgsqlsam (PostGreSQL-based backend) Makes use of a PostgreSQL database to store account information. This backend is largely undocumented at the moment, though its configuration is very similar to that of the mysqlsam backend.

xmlsam (XML-based datafile) Allows the account and password data to be stored in an XML format data file. This backend cannot be used for normal operation, it can only be used in conjunction with **pdbedit**'s pdb2pdb functionality. The Document Type Definition (DTD) file that is used might be subject to changes in the future. (See the XML reference[1] for a definition of XML terms.)

The *xmlsam* option can be useful for account migration between database backends or backups. Use of this tool allows the data to be edited before migration into another backend format.

10.2 Technical Information

Old Windows clients send plaintext passwords over the wire. Samba can check these passwords by encrypting them and comparing them to the hash stored in the UNIX user database.

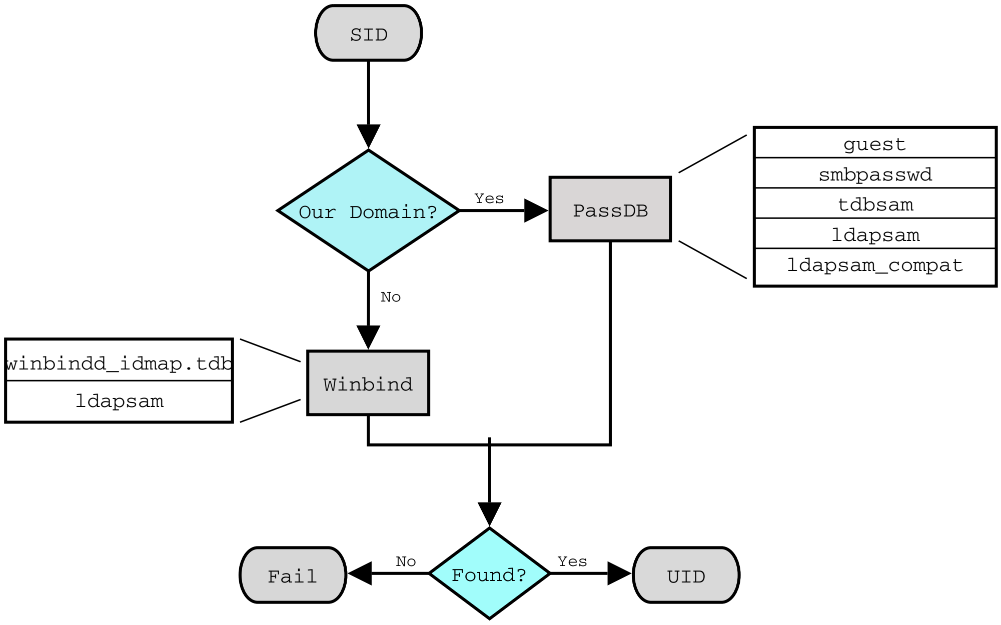
Newer Windows clients send encrypted passwords (LanMan and NT hashes) instead of plaintext passwords over the wire. The newest clients will send only encrypted passwords and refuse to send plaintext passwords unless their registry is tweaked.

Many people ask why Samba cannot simply use the UNIX password database. Windows requires passwords that are encrypted in its own format. The UNIX passwords can't be converted to UNIX-style encrypted passwords. Because of that, you can't use the standard UNIX user database, and you have to store the LanMan and NT hashes somewhere else.

[1] <`http://www.brics.dk/~amoeller/XML/schemas/`>

In addition to differently encrypted passwords, Windows also stores certain data for each user that is not stored in a UNIX user database: for example, workstations the user may logon from, the location where the user's profile is stored, and so on. Samba retrieves and stores this information using a *passdb backend*. Commonly available backends are LDAP, tdbsam, plain text file, and MySQL. For more information, see the man page for `smb.conf` regarding the *passdb backend* parameter.

Figure 10.1 IDMAP: Resolution of SIDs to UIDs.

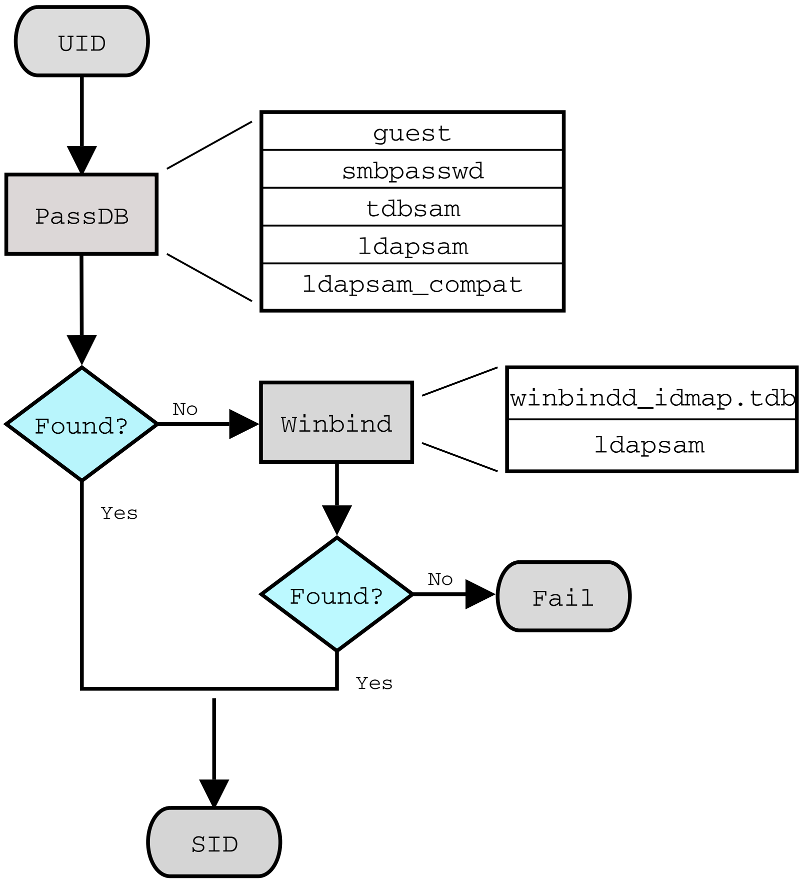


The resolution of SIDs to UIDs is fundamental to correct operation of Samba. In both cases shown, if winbindd is not running or cannot be contacted, then only local SID/UID resolution is possible. See Figure 10.1 and Figure 10.2 diagrams.

10.2.1 Important Notes About Security

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

Figure 10.2 IDMAP: Resolution of UIDs to SIDs.



Ideally, we would like a password scheme that involves neither plaintext passwords on the network nor plaintext passwords on disk. Unfortunately, this is not available because Samba is stuck with having to be compatible with other SMB systems (Windows NT, Windows for Workgroups, Windows 9x/Me).

Windows NT 4.0 Service Pack 3 changed the default setting so plaintext passwords are disabled from being sent over the wire. This mandates either the use of encrypted password support or editing the Windows NT registry to re-enable plaintext passwords.

The following versions of Microsoft Windows do not support full domain security protocols, although they may log onto a domain environment:

- MS DOS Network client 3.0 with the basic network redirector installed.
- Windows 95 with the network redirector update installed.
- Windows 98 [Second Edition].
- Windows Me.

Note

MS Windows XP Home does not have facilities to become a domain member, and it cannot participate in domain logons.

The following versions of MS Windows fully support domain security protocols.

- Windows NT 3.5x.
- Windows NT 4.0.
- Windows 2000 Professional.
- Windows 200x Server/Advanced Server.
- Windows XP Professional.

All current releases of Microsoft SMB/CIFS clients support authentication via the SMB challenge/response mechanism described here. Enabling clear-text authentication does not disable the ability of the client to participate in encrypted authentication. Instead, it allows the client to negotiate either plaintext or encrypted password handling.

MS Windows clients will cache the encrypted password alone. Where plaintext passwords are re-enabled through the appropriate registry change, the plaintext password is never cached. This means that in the event that a network connections should become disconnected (broken), only the cached (encrypted) password will be sent to the resource server to effect an auto-reconnect. If the resource server does not support encrypted passwords, the auto-reconnect will fail. Use of encrypted passwords is strongly advised.

10.2.1.1 Advantages of Encrypted Passwords

- Plaintext passwords are not passed across the network. Someone using a network sniffer cannot just record passwords going to the SMB server.
- Plaintext passwords are not stored anywhere in memory or on disk.
- Windows NT does not like talking to a server that does not support encrypted passwords. It will refuse to browse the server if the server is also in user-level security mode. It will insist on prompting the user for the password on each connection, which is very annoying. The only thing you can do to stop this is to use SMB encryption.
- Encrypted password support allows automatic share (resource) reconnects.
- Encrypted passwords are essential for PDC/BDC operation.

10.2.1.2 Advantages of Non-Encrypted Passwords

- Plaintext passwords are not kept on disk and are not cached in memory.
- Plaintext passwords use the same password file as other UNIX services, such as Login and FTP.
- Use of other services (such as Telnet and FTP) that send plaintext passwords over the network makes sending them for SMB not such a big deal.

10.2.2 Mapping User Identifiers between MS Windows and UNIX

Every operation in UNIX/Linux requires a user identifier (UID), just as in MS Windows NT4/200x this requires a security identifier (SID). Samba provides two means for mapping an MS Windows user to a UNIX/Linux UID.

First, all Samba SAM database accounts require a UNIX/Linux UID that the account will map to. As users are added to the account information database, Samba will call the *add user script* interface to add the account to the Samba host OS. In essence all accounts in the local SAM require a local user account.

The second way to map Windows SID to UNIX UID is via the *idmap uid* and *idmap gid* parameters in `smb.conf`. Please refer to the man page for information about these parameters. These parameters are essential when mapping users from a remote (non-member Windows client or a member of a foreign domain) SAM server.

10.2.3 Mapping Common UIDs/GIDs on Distributed Machines

Samba-3 has a special facility that makes it possible to maintain identical UIDs and GIDs on all servers in a distributed network. A distributed network is one where there exists a PDC, one or more BDCs, and/or one or more domain member servers. Why is this important? This is important if files are being shared over more than one protocol (e.g., NFS) and where users are copying files across UNIX/Linux systems using tools such as **rsync**.

The special facility is enabled using a parameter called *idmap backend*. The default setting for this parameter is an empty string. Technically it is possible to use an LDAP-based idmap backend for UIDs and GIDs, but it makes most sense when this is done for network configurations that also use LDAP for the SAM backend. Example 10.2.1 shows that configuration.

Example 10.2.1 Example Configuration with the LDAP idmap Backend

```
[global]
        idmap backend = ldap:ldap://ldap-server.quenya.org:636
# Alternatively, this could be specified as:
        idmap backend = ldap:ldaps://ldap-server.quenya.org
```

A network administrator who wants to make significant use of LDAP backends will sooner or later be exposed to the excellent work done by PADL Software. PADL <`http://www.padl.com`> have produced and released to open source an array of tools that might be of interest. These tools include:

- *nss_ldap:* An LDAP name service switch (NSS) module to provide native name service support for AIX, Linux, Solaris, and other operating systems. This tool can be used for centralized storage and retrieval of UIDs and GIDs.
- *pam_ldap:* A PAM module that provides LDAP integration for UNIX/Linux system access authentication.
- *idmap_ad:* An IDMAP backend that supports the Microsoft Services for UNIX RFC 2307 schema available from the PADL Web site[2].

10.2.4 Comments Regarding LDAP

There is much excitement and interest in LDAP directories in the information technology world today. The LDAP architecture was designed to be highly scalable. It was also designed for use across a huge number of potential areas of application encompassing a wide range of operating systems and platforms. LDAP technologies are at the heart of the current generations of Federated Identity Management (FIM) solutions that can underlie a corporate Single Sign-On (SSO) environment.

LDAP implementations have been built across a wide variety of platforms. It lies at the core of Microsoft Windows Active Directory services (ADS), Novell's eDirectory, as well as many others. Implementation of the directory services LDAP involves interaction with legacy as well as new generation applications, all of which depend on some form of authentication services.

UNIX services can utilize LDAP directory information for authentication and access controls through intermediate tools and utilities. The total environment that consists of the LDAP directory and the middle-ware tools and utilities makes it possible for all user access to the UNIX platform to be managed from a central environment and yet distributed to wherever the point of need may be physically located. Applications that benefit from this infrastructure include: UNIX login shells, mail and messaging systems, quota controls, printing systems, DNS servers, DHCP servers, and also Samba.

Many sites are installing LDAP for the first time in order to provide a scalable passdb backend for Samba. Others are faced with the need to adapt an existing LDAP directory to new uses such as for the Samba SAM backend. Whatever your particular need and attraction to Samba may be, decisions made in respect of the design of the LDAP directory structure and its implementation are of a durable nature for the site. These have far-reaching implications that affect long-term information systems management costs.

Do not rush into an LDAP deployment. Take the time to understand how the design of the Directory Information Tree (DIT) may impact current and future site needs, as well as the ability to meet them. The way that Samba SAM information should be stored within

[2] <`http://www.padl.com/download/xad_oss_plugins.tar.gz`>

the DIT varies from site to site and with each implementation new experience is gained. It is well understood by LDAP veterans that first implementations create awakening, second implementations of LDAP create fear, and third-generation deployments bring peace and tranquility.

10.2.4.1 Caution Regarding LDAP and Samba

Samba requires UNIX POSIX identity information as well as a place to store information that is specific to Samba and the Windows networking environment. The most used information that must be dealt with includes: user accounts, group accounts, machine trust accounts, interdomain trust accounts, and intermediate information specific to Samba internals.

The example deployment guidelines in this book, as well as other books and HOWTO documents available from the internet may not fit with established directory designs and implementations. The existing DIT may not be able to accommodate the simple information layout proposed in common sources. Additionally, you may find that the common scripts and tools that are used to provision the LDAP directory for use with Samba may not suit your needs.

It is not uncommon, for sites that have existing LDAP DITs to find necessity to generate a set of site-specific scripts and utilities to make it possible to deploy Samba within the scope of site operations. The way that user and group accounts are distributed throughout the DIT may make this a challenging matter. The solution will, of course, be rewarding, but the journey to it may be challenging. Take time to understand site needs and do not rush into deployment.

Above all, do not blindly use scripts and tools that are not suitable for your site. Check and validate all scripts before you execute them to make sure that the existing infrastructure will not be damaged by inadvertent use of an inappropriate tool.

10.2.5 LDAP Directories and Windows Computer Accounts

Samba doesn't provide a turnkey solution to LDAP. It is best to deal with the design and configuration of an LDAP directory prior to integration with Samba. A working knowledge of LDAP makes Samba integration easy, and the lack of a working knowledge of LDAP can make it a frustrating experience.

Computer (machine) accounts can be placed wherever you like in an LDAP directory subject to some constraints that are described in this chapter.

The POSIX and sambaSamAccount components of computer (machine) accounts are both used by Samba. Thus, machine accounts are treated inside Samba in the same way that Windows NT4/200X treats them. A user account and a machine account are indistinquishable from each other, except that the machine account ends in a $ character, as do trust accounts.

The need for Windows user, group, machine, trust, and other accounts to be tied to a valid UNIX UID is a design decision that was made a long way back in the history of

Samba development. It is unlikely that this decision will be reversed or changed during the remaining life of the Samba-3.x series.

The resolution of a UID from the Windows SID is achieved within Samba through a mechanism that must refer back to the host operating system on which Samba is running. The NSS is the preferred mechanism that shields applications (like Samba) from the need to know everything about every host OS it runs on.

Samba asks the host OS to provide a UID via the "passwd", "shadow", and "group" facilities in the NSS control (configuration) file. The best tool for achieving this is left up to the UNIX administrator to determine. It is not imposed by Samba. Samba provides winbindd with its support libraries as one method. It is possible to do this via LDAP, and for that Samba provides the appropriate hooks so that all account entities can be located in an LDAP directory.

For many the weapon of choice is to use the PADL nss_ldap utility. This utility must be configured so that computer accounts can be resolved to a POSIX/UNIX account UID. That is fundamentally an LDAP design question. The information provided on the Samba list and in the documentation is directed at providing working examples only. The design of an LDAP directory is a complex subject that is beyond the scope of this documentation.

10.3 Account Management Tools

Samba provides two tools for management of user and machine accounts: **smbpasswd** and **pdbedit**.

The **pdbedit** can be used to manage account policies in addition to Samba user account information. The policy management capability is used to administer domain default settings for password aging and management controls to handle failed login attempts.

Some people are confused when reference is made to `smbpasswd` because the name refers to a storage mechanism for SambaSAMAccount information, but it is also the name of a utility tool. That tool is destined to eventually be replaced by new functionality that is being added to the **net** toolset (see Chapter 12, "Remote and Local Management: The Net Command".

10.3.1 The smbpasswd Tool

The **smbpasswd** utility is similar to the **passwd** and **yppasswd** programs. It maintains the two 32 byte password fields in the passdb backend. This utility operates independently of the actual account and password storage methods used (as specified by the ***passdb backend*** in the `smb.conf` file.

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

smbpasswd has the capability to change passwords on Windows NT servers (this only works when the request is sent to the NT PDC if changing an NT domain user's password).

smbpasswd can be used to:

- *add* user or machine accounts.
- *delete* user or machine accounts.
- *enable* user or machine accounts.
- *disable* user or machine accounts.
- *set to NULL* user passwords.
- *manage* interdomain trust accounts.

To run smbpasswd as a normal user, just type:

```
$ smbpasswd
Old SMB password: secret
```

For *secret*, type the old value here or press return if there is no old password.

```
New SMB Password: new secret
Repeat New SMB Password: new secret
```

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

When invoked by an ordinary user, the command will allow only the user to change his or her own SMB password.

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

smbpasswd is designed to work in the way familiar to UNIX users who use the **passwd** or **yppasswd** commands. While designed for administrative use, this tool provides essential user-level password change capabilities.

For more details on using **smbpasswd**, refer to the man page (the definitive reference).

10.3.2 The pdbedit Tool

pdbedit is a tool that can be used only by root. It is used to manage the passdb backend, as well as domain-wide account policy settings. **pdbedit** can be used to:

- add, remove, or modify user accounts.
- list user accounts.
- migrate user accounts.

- migrate group accounts.
- manage account policies.
- manage domain access policy settings.

Under the terms of the Sarbanes-Oxley Act of 2002, American businesses and organizations are mandated to implement a series of `internal controls` and procedures to communicate, store, and protect financial data. The Sarbanes-Oxley Act has far reaching implications in respect of:

1. Who has access to information systems that store financial data.
2. How personal and finacial information is treated among employees and business partners.
3. How security vulnerabilities are managed.
4. Security and patch level maintenance for all information systems.
5. How information systems changes are documented and tracked.
6. How information access controls are implemented and managed.
7. Auditability of all information systems in respect of change and security.
8. Disciplinary procedures and controls to ensure privacy.

In short, the Sarbanes-Oxley Act of 2002 is an instrument that enforces accountability in respect of business related information systems so as to ensure the compliance of all information systems that are used to store personal information and particularly for financial records processing. Similar accountabilities are being demanded around the world.

The need to be familiar with the Samba tools and facilities that permit information systems operation in compliance with government laws and regulations is clear to all. The **pdbedit** is currently the only Samba tool that provides the capacity to manage account and systems access controls and policies. During the remaining life-cycle of the Samba-3 series it is possible the new tools may be implemented to aid in this important area.

Domain global policy controls available in Windows NT4 compared with Samba is shown in Table 10.1.

The **pdbedit** tool is the only one that can manage the account security and policy settings. It is capable of all operations that smbpasswd can do as well as a superset of them.

One particularly important purpose of the **pdbedit** is to allow the migration of account information from one passdb backend to another. See the Section 10.4.6 password backend section of this chapter.

10.3.2.1 User Account Management

The **pdbedit** tool, like the **smbpasswd** tool, requires that a POSIX user account already exists in the UNIX/Linux system accounts database (backend). Neither tool will call out

Table 10.1 NT4 Domain v's Samba Policy Controls

NT4 policy Name	**Samba Policy Name**	**NT4 Range**	**Samba Range**	**Samba Default**
Maximum Password Age	maximum password age	0 - 999 (days)	0 - 4294967295 (sec)	4294967295
Minimum Password Age	minimum password age	0 - 999 (days)	0 - 4294967295 (sec)	0
Mimimum Password Length	min password length	1 - 14 (Chars)	0 - 4294967295 (Chars)	5
Password Uniqueness	password history	0 - 23 (#)	0 - 4294967295 (#)	0
Account Lockout - Reset count after	reset count minutes	1 - 99998 (min)	0 - 4294967295 (min)	30
Lockout after bad logon attempts	bad lockout attempt	0 - 998 (#)	0 - 4294967295 (#)	0
*** Not Known ***	disconnect time	TBA	0 - 4294967295	0
Lockout Duration	lockout duration	1 - 99998 (min)	0 - 4294967295 (min)	30
Users must log on in order to change password	user must logon to change password	0/1	0 - 4294967295	0
*** Registry Setting ***	refuse machine password change	0/1	0 - 4294967295	0

to the operating system to create a user account because this is considered to be the responsibility of the system administrator. When the Windows NT4 domain user manager is used to add an account, Samba will implement the `add user script` (as well as the other interface scripts) to ensure that user, group and machine accounts are correctly created and changed. The use of the **pdbedit** tool does not make use of these interface scripts.

Before attempting to use the **pdbedit** tool to manage user and machine accounts, make certain that a system (POSIX) account has already been created.

Listing User and Machine Accounts The following is an example of the user account information that is stored in a tdbsam password backend. This listing was produced by running:

```
$ pdbedit -Lv met
UNIX username:        met
NT username:          met
Account Flags:        [U          ]
User SID:             S-1-5-21-1449123459-1407424037-3116680435-2004
Primary Group SID:    S-1-5-21-1449123459-1407424037-3116680435-1201
Full Name:            Melissa E Terpstra
Home Directory:       \\frodo\met\Win9Profile
HomeDir Drive:        H:
Logon Script:         scripts\logon.bat
Profile Path:         \\frodo\Profiles\met
Domain:               MIDEARTH
Account desc:
Workstations:         melbelle
Munged dial:
Logon time:           0
Logoff time:          Mon, 18 Jan 2038 20:14:07 GMT
Kickoff time:         Mon, 18 Jan 2038 20:14:07 GMT
Password last set:    Sat, 14 Dec 2002 14:37:03 GMT
Password can change:  Sat, 14 Dec 2002 14:37:03 GMT
Password must change: Mon, 18 Jan 2038 20:14:07 GMT
```

Accounts can also be listed in the older `smbpasswd` format:

```
root# pdbedit -Lw
root:0:84B0D8E14D158FF8417EAF50CFAC29C3:
     AF6DD3FD4E2EA8BDE1695A3F05EFBF52:[U          ]:LCT-42681AB8:
jht:1000:6BBC4159020A52741486235A2333E4D2:
     CC099521AD554A3C3CF2556274DBCFBC:[U          ]:LCT-40D75B5B:
rcg:1002:E95D4331A6F23AF8AAD3B435B51404EE:
     BB0F2C39B04CA6100F0E535DF8314B43:[U          ]:LCT-40D7C5A3:
afw:1003:1AAFA7F9F6DC1DEAAAD3B435B51404EE:
     CE92C2F9471594CDC4E7860CA6BC62DB:[T          ]:LCT-40DA501F:
met:1004:A2848CB7E076B435AAD3B435B51404EE:
     F25F5D3405085C555236B80B7B22C0D2:[U          ]:LCT-4244FAB8:
aurora$:1005:060DE593EA638B8ACC4A19F14D2FF2BB:
     060DE593EA638B8ACC4A19F14D2FF2BB:[W          ]:LCT-4173E5CC:
temptation$:1006:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX:
     A96703C014E404E33D4049F706C45EE9:[W          ]:LCT-42BF0C57:
vaioboss$:1001:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX:
     88A30A095160072784C88F811E89F98A:[W          ]:LCT-41C3878D:
frodo$:1008:15891DC6B843ECA41249940C814E316B:
     B68EADCCD18E17503D3DAD3E6B0B9A75:[W          ]:LCT-42B7979F:
marvel$:1011:BF709959C3C94E0B3958B7B84A3BB6F3:
     C610EFE9A385A3E8AA46ADFD576E6881:[W          ]:LCT-40F07A4
```

Adding User Accounts The **pdbedit** can be used to add a user account to a standalone server or to a domain. In the example shown here the account for the user `vlaan` has been created before attempting to add the SambaSAMAccount.

```
root#  pdbedit -a vlaan
new password: secretpw
retype new password: secretpw
Unix username:        vlaan
NT username:          vlaan
Account Flags:        [U          ]
User SID:             S-1-5-21-726309263-4128913605-1168186429-3014
Primary Group SID:    S-1-5-21-726309263-4128913605-1168186429-513
Full Name:            Victor Laan
Home Directory:       \\frodo\vlaan
HomeDir Drive:        H:
Logon Script:         scripts\logon.bat
Profile Path:         \\frodo\profiles\vlaan
Domain:               MIDEARTH
Account desc:         Guest User
Workstations:
Munged dial:
Logon time:           0
Logoff time:          Mon, 18 Jan 2038 20:14:07 GMT
Kickoff time:         Mon, 18 Jan 2038 20:14:07 GMT
Password last set:    Wed, 29 Jun 2005 19:35:12 GMT
Password can change:  Wed, 29 Jun 2005 19:35:12 GMT
Password must change: Mon, 18 Jan 2038 20:14:07 GMT
Last bad password   : 0
Bad password count  : 0
Logon hours         : FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
```

Deleting Accounts An account can be deleted from the SambaSAMAccount database

```
root#  pdbedit -x vlaan
```

The account is removed without further screen output. The account is removed only from the SambaSAMAccount (passdb backend) database, it is not removed from the UNIX account backend.

The use of the NT4 domain user manager to delete an account will trigger the *`delete user script`*, but not the **pdbedit** tool.

Changing User Accounts Refer to the **pdbedit** man page for a full synopsis of all operations that are available with this tool.

An example of a simple change in the user account information is the change of the full name information shown here:

```
root#  pdbedit -r --fullname="Victor Aluicious Laan" vlaan
...
Primary Group SID:    S-1-5-21-726309263-4128913605-1168186429-513
Full Name:            Victor Aluicious Laan
Home Directory:       \\frodo\vlaan
...
```

Let us assume for a moment that a user's password has expired and the user is unable to change the password at this time. It may be necessary to give the user additional grace time so that it is possible to continue to work with the account and the original password. This demonstrates how the password expiration settings may be updated

```
root#  pdbedit -Lv vlaan
...
Password last set:    Sun, 09 Sep 2001 22:21:40 GMT
Password can change:  Thu, 03 Jan 2002 15:08:35 GMT
Password must change: Thu, 03 Jan 2002 15:08:35 GMT
Last bad password   : Thu, 03 Jan 2002 15:08:35 GMT
Bad password count  : 2
...
```

The user has recorded 2 bad logon attempts and the next will lock the account, but the password is also expired. Here is how this account can be reset:

```
root#  pdbedit -z vlaan
...
Password last set:    Sun, 09 Sep 2001 22:21:40 GMT
Password can change:  Thu, 03 Jan 2002 15:08:35 GMT
Password must change: Thu, 03 Jan 2002 15:08:35 GMT
Last bad password   : 0
Bad password count  : 0
...
```

The `Password must change:` parameter can be reset like this:

```
root#  pdbedit --pwd-must-change-time=1200000000 vlaan
...
Password last set:    Sun, 09 Sep 2001 22:21:40 GMT
Password can change:  Thu, 03 Jan 2002 15:08:35 GMT
Password must change: Thu, 10 Jan 2008 14:20:00 GMT
...
```

Another way to use this tools is to set the date like this:

```
root#  pdbedit --pwd-must-change-time="2010-01-01" \
              --time-format="%Y-%m-%d" vlaan
...
Password last set:    Sun, 09 Sep 2001 22:21:40 GMT
Password can change:  Thu, 03 Jan 2002 15:08:35 GMT
Password must change: Fri, 01 Jan 2010 00:00:00 GMT
...
```

Refer to the strptime man page for specific time format information.

Please refer to the pdbedit man page for further information relating to SambaSAMAccount management.

Domain Account Policy Managment To view the domain account access policies that may be configured execute:

```
root#  pdbedit -P ?
No account policy by that name
Account policy names are :
min password length
password history
user must logon to change password
maximum password age
minimum password age
lockout duration
reset count minutes
bad lockout attempt
disconnect time
refuse machine password change
```

Commands will be executed to establish controls for our domain as follows:

1. min password length = 8 characters.
2. password history = last 4 passwords.
3. maximum password age = 90 days.
4. minimum password age = 7 days.
5. bad lockout attempt = 8 bad logon attempts.

6. lockout duration = forever, account must be manually reenabled.

The following command execution will achieve these settings:

```
root#  pdbedit -P "min password length" -C 8
account policy value for min password length was 5
account policy value for min password length is now 8
root#  pdbedit -P "password history" -C 4
account policy value for password history was 0
account policy value for password history is now 4
root#  pdbedit -P "maximum password age" -C 90
account policy value for maximum password age was 4294967295
account policy value for maximum password age is now 90
root#  pdbedit -P "minimum password age" -C 7
account policy value for minimum password age was 0
account policy value for minimum password age is now 7
root#  pdbedit -P "bad lockout attempt" -C 8
account policy value for bad lockout attempt was 0
account policy value for bad lockout attempt is now 8
root#  pdbedit -P "lockout duration" -C -1
account policy value for lockout duration was 30
account policy value for lockout duration is now 4294967295
```

NOTE

To set the maximum (infinite) lockout time use the value of -1.

WARNING

Account policies must be set individually on each PDC and BDC. At this time (Samba 3.0.11 to Samba 3.0.14a) account policies are not replicated automatically. This may be fixed before Samba 3.0.20 ships or some time there after.

10.3.2.2 Account Migration

The **pdbedit** tool allows migration of authentication (account) databases from one backend to another. For example, to migrate accounts from an old `smbpasswd` database to a *`tdbsam`* backend:

1. Set the *passdb backend* = tdbsam, smbpasswd.
2. Execute:

   ```
   root# pdbedit -i smbpasswd -e tdbsam
   ```

3. Remove the *smbpasswd* from the passdb backend configuration in `smb.conf`.

10.4 Password Backends

Samba offers the greatest flexibility in backend account database design of any SMB/CIFS server technology available today. The flexibility is immediately obvious as one begins to explore this capability.

It is possible to specify not only multiple password backends, but even multiple backends of the same type. For example, to use two different `tdbsam` databases:

```
        passdb backend = tdbsam:/etc/samba/passdb.tdb tdbsam:/etc/ ↩
            samba/old-passdb.tdb
```

What is possible is not always sensible. Be careful to avoid complexity to the point that it may be said that the solution is “too clever by half!”

10.4.1 Plaintext

Older versions of Samba retrieved user information from the UNIX user database and eventually some other fields from the file `/etc/samba/smbpasswd` or `/etc/smbpasswd`. When password encryption is disabled, no SMB-specific data is stored at all. Instead, all operations are conducted via the way that the Samba host OS will access its `/etc/passwd` database. On most Linux systems, for example, all user and group resolution is done via PAM.

10.4.2 smbpasswd: Encrypted Password Database

Traditionally, when configuring *encrypt passwords* = yes in Samba’s `smb.conf` file, user account information such as username, LM/NT password hashes, password change times, and account flags have been stored in the `smbpasswd(5)` file. There are several disadvantages to this approach for sites with large numbers of users (counted in the thousands).

- The first problem is that all lookups must be performed sequentially. Given that there are approximately two lookups per domain logon (one during intial logon validation and one for a session connection setup, such as when mapping a network drive or printer), this is a performance bottleneck for large sites. What is needed is an indexed approach such as that used in databases.

- The second problem is that administrators who desire to replicate an smbpasswd file to more than one Samba server are left to use external tools such as **rsync(1)** and **ssh(1)** and write custom, in-house scripts.
- Finally, the amount of information that is stored in an smbpasswd entry leaves no room for additional attributes such as a home directory, password expiration time, or even a relative identifier (RID).

As a result of these deficiencies, a more robust means of storing user attributes used by smbd was developed. The API that defines access to user accounts is commonly referred to as the samdb interface (previously, this was called the passdb API and is still so named in the Samba source code trees).

Samba provides an enhanced set of passdb backends that overcome the deficiencies of the smbpasswd plaintext database. These are tdbsam, ldapsam, and xmlsam. Of these, ldapsam will be of most interest to large corporate or enterprise sites.

10.4.3 tdbsam

Samba can store user and machine account data in a "TDB" (trivial database). Using this backend does not require any additional configuration. This backend is recommended for new installations that do not require LDAP.

As a general guide, the Samba Team does not recommend using the tdbsam backend for sites that have 250 or more users. Additionally, tdbsam is not capable of scaling for use in sites that require PDB/BDC implementations that require replication of the account database. Clearly, for reason of scalability, the use of ldapsam should be encouraged.

The recommendation of a 250-user limit is purely based on the notion that this would generally involve a site that has routed networks, possibly spread across more than one physical location. The Samba Team has not at this time established the performance-based scalability limits of the tdbsam architecture.

There are sites that have thousands of users and yet require only one server. One site recently reported having 4,500 user accounts on one UNIX system and reported excellent performance with the `tdbsam` passdb backend. The limitation of where the `tdbsam` passdb backend can be used is not one pertaining to a limitation in the TDB storage system, it is based only on the need for a reliable distribution mechanism for the SambaSAMAccount backend.

10.4.4 ldapsam

There are a few points to stress that the ldapsam does not provide. The LDAP support referred to in this documentation does not include:

- A means of retrieving user account information from a Windows 200x Active Directory server.
- A means of replacing /etc/passwd.

The second item can be accomplished by using LDAP NSS and PAM modules. LGPL versions of these libraries can be obtained from PADL Software[3]. More information about the configuration of these packages may be found in *LDAP, System Administration* by Gerald Carter, Chapter 6, Replacing NIS"[4].

This document describes how to use an LDAP directory for storing Samba user account information traditionally stored in the smbpasswd(5) file. It is assumed that the reader already has a basic understanding of LDAP concepts and has a working directory server already installed. For more information on LDAP architectures and directories, please refer to the following sites:

- OpenLDAP[5]
- Sun One Directory Server[6]
- Novell eDirectory[7]
- IBM Tivoli Directory Server[8]
- Red Hat Directory Server[9]
- Fedora Directory Server[10]

Two additional Samba resources that may prove to be helpful are:

- The Samba-PDC-LDAP-HOWTO[11] maintained by Ignacio Coupeau.
- The NT migration scripts from IDEALX[12] that are geared to manage users and groups in such a Samba-LDAP domain controller configuration. Idealx also produced the smbldap-tools and the Interactive Console Management tool.

10.4.4.1 Supported LDAP Servers

The LDAP ldapsam code was developed and tested using the OpenLDAP 2.x server and client libraries. The same code should work with Netscape's Directory Server and client SDK. However, there are bound to be compile errors and bugs. These should not be hard to fix. Please submit fixes via the process outlined in Chapter 39, "Reporting Bugs".

Samba is capable of working with any standards-compliant LDAP server.

[3]<http://www.padl.com/>
[4]<http://safari.oreilly.com/?XmlId=1-56592-491-6>
[5]<http://www.openldap.org/>
[6]<http://www.sun.com/software/products/directory_srvr_ee/index.xml>
[7]<http://www.novell.com/products/edirectory/>
[8]<http://www-306.ibm.com/software/tivoli/products/directory-server/>
[9]<http://www.redhat.com/software/rha/directory/>
[10]<http://www.linuxsecurity.com/content/view/119229>
[11]<http://www.unav.es/cti/ldap-smb/ldap-smb-3-howto.html>
[12]<http://samba.idealx.org/>

10.4.4.2 Schema and Relationship to the RFC 2307 posixAccount

Samba-3.0 includes the necessary schema file for OpenLDAP 2.x in the `examples/LDAP/samba.schema` directory of the source code distribution tarball. The schema entry for the sambaSamAccount ObjectClass is shown here:

```
ObjectClass (1.3.6.1.4.1.7165.2.2.6 NAME 'sambaSamAccount' SUP top AUXILIARY
    DESC 'Samba-3.0 Auxiliary SAM Account'
    MUST ( uid $ sambaSID )
    MAY  ( cn $ sambaLMPassword $ sambaNTPassword $ sambaPwdLastSet $
          sambaLogonTime $ sambaLogoffTime $ sambaKickoffTime $
          sambaPwdCanChange $ sambaPwdMustChange $ sambaAcctFlags $
          displayName $ sambaHomePath $ sambaHomeDrive $ sambaLogonScript $
          sambaProfilePath $ description $ sambaUserWorkstations $
          sambaPrimaryGroupSID $ sambaDomainName ))
```

The `samba.schema` file has been formatted for OpenLDAP 2.0/2.1. The Samba Team owns the OID space used by the above schema and recommends its use. If you translate the schema to be used with Netscape DS, please submit the modified schema file as a patch to <`jerry@samba.org`>[13].

Just as the smbpasswd file is meant to store information that provides information additional to a user's `/etc/passwd` entry, so is the sambaSamAccount object meant to supplement the UNIX user account information. A sambaSamAccount is an `AUXILIARY` ObjectClass, so it can be used to augment existing user account information in the LDAP directory, thus providing information needed for Samba account handling. However, there are several fields (e.g., uid) that overlap with the posixAccount ObjectClass outlined in RFC 2307. This is by design.

In order to store all user account information (UNIX and Samba) in the directory, it is necessary to use the sambaSamAccount and posixAccount ObjectClasses in combination. However, **smbd** will still obtain the user's UNIX account information via the standard C library calls, such as getpwnam(). This means that the Samba server must also have the LDAP NSS library installed and functioning correctly. This division of information makes it possible to store all Samba account information in LDAP, but still maintain UNIX account information in NIS while the network is transitioning to a full LDAP infrastructure.

10.4.4.3 OpenLDAP Configuration

To include support for the sambaSamAccount object in an OpenLDAP directory server, first copy the samba.schema file to slapd's configuration directory. The samba.schema file can be found in the directory `examples/LDAP` in the Samba source distribution.

[13]<`mailto:jerry@samba.org`>

```
root# cp samba.schema /etc/openldap/schema/
```

Next, include the `samba.schema` file in `slapd.conf`. The sambaSamAccount object contains two attributes that depend on other schema files. The ***uid*** attribute is defined in `cosine.schema` and the ***displayName*** attribute is defined in the `inetorgperson.schema` file. Both of these must be included before the `samba.schema` file.

```
## /etc/openldap/slapd.conf

## schema files (core.schema is required by default)
include             /etc/openldap/schema/core.schema

## needed for sambaSamAccount
include            /etc/openldap/schema/cosine.schema
include            /etc/openldap/schema/inetorgperson.schema
include            /etc/openldap/schema/nis.schema
include            /etc/openldap/schema/samba.schema
....
```

It is recommended that you maintain some indices on some of the most useful attributes, as in the following example, to speed up searches made on sambaSamAccount ObjectClasses (and possibly posixAccount and posixGroup as well):

```
# Indices to maintain
## required by OpenLDAP
index objectclass             eq

index cn                      pres,sub,eq
index sn                      pres,sub,eq
## required to support pdb_getsampwnam
index uid                     pres,sub,eq
## required to support pdb_getsambapwrid()
index displayName             pres,sub,eq

## uncomment these if you are storing posixAccount and
## posixGroup entries in the directory as well
##index uidNumber               eq
##index gidNumber               eq
##index memberUid               eq

index   sambaSID              eq
index   sambaPrimaryGroupSID  eq
index   sambaDomainName       eq
index   default              sub
```

Create the new index by executing:

```
root# ./sbin/slapindex -f slapd.conf
```

Remember to restart slapd after making these changes:

```
root# /etc/init.d/slapd restart
```

10.4.4.4 Initialize the LDAP Database

Before you can add accounts to the LDAP database, you must create the account containers that they will be stored in. The following LDIF file should be modified to match your needs (DNS entries, and so on):

```
# Organization for Samba Base
dn: dc=quenya,dc=org
objectclass: dcObject
objectclass: organization
dc: quenya
o: Quenya Org Network
description: The Samba-3 Network LDAP Example

# Organizational Role for Directory Management
dn: cn=Manager,dc=quenya,dc=org
objectclass: organizationalRole
cn: Manager
description: Directory Manager

# Setting up container for Users OU
dn: ou=People,dc=quenya,dc=org
objectclass: top
objectclass: organizationalUnit
ou: People

# Setting up admin handle for People OU
dn: cn=admin,ou=People,dc=quenya,dc=org
cn: admin
objectclass: top
objectclass: organizationalRole
objectclass: simpleSecurityObject
userPassword: {SSHA}c3ZM9tBaBo9autm1dL3waDS21+JSfQVz
```

```
# Setting up container for groups
dn: ou=Groups,dc=quenya,dc=org
objectclass: top
objectclass: organizationalUnit
ou: Groups

# Setting up admin handle for Groups OU
dn: cn=admin,ou=Groups,dc=quenya,dc=org
cn: admin
objectclass: top
objectclass: organizationalRole
objectclass: simpleSecurityObject
userPassword: {SSHA}c3ZM9tBaBo9autm1dL3waDS21+JSfQVz

# Setting up container for computers
dn: ou=Computers,dc=quenya,dc=org
objectclass: top
objectclass: organizationalUnit
ou: Computers

# Setting up admin handle for Computers OU
dn: cn=admin,ou=Computers,dc=quenya,dc=org
cn: admin
objectclass: top
objectclass: organizationalRole
objectclass: simpleSecurityObject
userPassword: {SSHA}c3ZM9tBaBo9autm1dL3waDS21+JSfQVz
```

The userPassword shown above should be generated using **slappasswd**.

The following command will then load the contents of the LDIF file into the LDAP database.

```
$ slapadd -v -l initldap.dif
```

Do not forget to secure your LDAP server with an adequate access control list as well as an admin password.

NOTE

Before Samba can access the LDAP server, you need to store the LDAP admin password in the Samba-3 `secrets.tdb` database by:

```
root# smbpasswd -w secret
```

10.4.4.5 Configuring Samba

The following parameters are available in `smb.conf` only if your version of Samba was built with LDAP support. Samba automatically builds with LDAP support if the LDAP libraries are found. The best method to verify that Samba was built with LDAP support is:

```
root#  smbd -b | grep LDAP
   HAVE_LDAP_H
   HAVE_LDAP
   HAVE_LDAP_DOMAIN2HOSTLIST
   HAVE_LDAP_INIT
   HAVE_LDAP_INITIALIZE
   HAVE_LDAP_SET_REBIND_PROC
   HAVE_LIBLDAP
   LDAP_SET_REBIND_PROC_ARGS
```

If the build of the **smbd** command you are using does not produce output that includes `HAVE_LDAP_H` it is necessary to discover why the LDAP headers and libraries were not found during compilation.

LDAP-related smb.conf options include these:

```
        passdb backend = ldapsam:url
        ldap admin dn
        ldap delete dn
        ldap filter
        ldap group suffix
        ldap idmap suffix
        ldap machine suffix
        ldap passwd sync
        ldap ssl
        ldap suffix
        ldap user suffix
        ldap replication sleep
        ldap timeout
        ldap page size
```

These are described in the `smb.conf` man page and so are not repeated here. However, an example for use with an LDAP directory is shown in Example 10.4.1

Example 10.4.1 Configuration with LDAP

```
[global]
        security = user
        encrypt passwords = yes
        netbios name = MORIA
        workgroup = NOLDOR
# LDAP related parameters:
# Define the DN used when binding to the LDAP servers.
# The password for this DN is not stored in smb.conf
# Set it using 'smbpasswd -w secret' to store the
# passphrase in the secrets.tdb file.
# If the "ldap admin dn" value changes, it must be reset.
        ldap admin dn = "cn=Manager,dc=quenya,dc=org"
# SSL directory connections can be configured by:
# ('off', 'start tls', or 'on' (default))
        ldap ssl = start tls
# syntax: passdb backend = ldapsam:ldap://server-name[:port]
        passdb backend = ldapsam:ldap://frodo.quenya.org
# smbpasswd -x delete the entire dn-entry
        ldap delete dn = no
# The machine and user suffix are added to the base suffix
# wrote WITHOUT quotes. NULL suffixes by default
        ldap user suffix = ou=People
        ldap group suffix = ou=Groups
        ldap machine suffix = ou=Computers
# Trust UNIX account information in LDAP
# (see the smb.conf man page for details)
# Specify the base DN to use when searching the directory
        ldap suffix = dc=quenya,dc=org
```

10.4.4.6 Accounts and Groups Management

Because user accounts are managed through the sambaSamAccount ObjectClass, you should modify your existing administration tools to deal with sambaSamAccount attributes.

Machine accounts are managed with the sambaSamAccount ObjectClass, just like user accounts. However, it is up to you to store those accounts in a different tree of your LDAP namespace. You should use "ou=Groups,dc=quenya,dc=org" to store groups and "ou=People,dc=quenya,dc=org" to store users. Just configure your NSS and PAM accordingly (usually, in the `/etc/openldap/sldap.conf` configuration file).

In Samba-3, the group management system is based on POSIX groups. This means that Samba makes use of the posixGroup ObjectClass. For now, there is no NT-like group system management (global and local groups). Samba-3 knows only about `Domain Groups` and, unlike MS Windows 2000 and Active Directory, Samba-3 does not support nested groups.

10.4.4.7 Security and sambaSamAccount

There are two important points to remember when discussing the security of sambaSAMAccount entries in the directory.

- *Never* retrieve the SambaLMPassword or SambaNTPassword attribute values over an unencrypted LDAP session.
- *Never* allow non-admin users to view the SambaLMPassword or SambaNTPassword attribute values.

These password hashes are clear-text equivalents and can be used to impersonate the user without deriving the original clear-text strings. For more information on the details of LM/NT password hashes, refer to Chapter 10, "Account Information Databases".

To remedy the first security issue, the *ldap ssl* `smb.conf` parameter defaults to require an encrypted session (*ldap ssl* = on) using the default port of `636` when contacting the directory server. When using an OpenLDAP server, it is possible to use the StartTLS LDAP extended operation in the place of LDAPS. In either case, you are strongly encouraged to use secure communications protocols (so do not set *ldap ssl* = off).

Note that the LDAPS protocol is deprecated in favor of the LDAPv3 StartTLS extended operation. However, the OpenLDAP library still provides support for the older method of securing communication between clients and servers.

The second security precaution is to prevent non-administrative users from harvesting password hashes from the directory. This can be done using the following ACL in `slapd.conf`:

```
## allow the "ldap admin dn" access, but deny everyone else
access to attrs=SambaLMPassword,SambaNTPassword
     by dn="cn=Samba Admin,ou=People,dc=quenya,dc=org" write
     by * none
```

10.4.4.8 LDAP Special Attributes for sambaSamAccounts

The sambaSamAccount ObjectClass is composed of the attributes shown in next tables: Table 10.2, and Table 10.3.

The majority of these parameters are only used when Samba is acting as a PDC of a domain (refer to Chapter 4, "Domain Control", for details on how to configure Samba as a PDC). The following four attributes are only stored with the sambaSamAccount entry if the values are non-default values:

- sambaHomePath
- sambaLogonScript

- sambaProfilePath
- sambaHomeDrive

These attributes are only stored with the sambaSamAccount entry if the values are non-default values. For example, assume MORIA has now been configured as a PDC and that *logon home* = \\%L\%u was defined in its `smb.conf` file. When a user named "becky" logs on to the domain, the *logon home* string is expanded to \\MORIA\becky. If the smbHome attribute exists in the entry "uid=becky,ou=People,dc=samba,dc=org", this value is used. However, if this attribute does not exist, then the value of the *logon home* parameter is used in its place. Samba will only write the attribute value to the directory entry if the value is something other than the default (e.g., `\\MOBY\becky`).

10.4.4.9 Example LDIF Entries for a sambaSamAccount

The following is a working LDIF that demonstrates the use of the SambaSamAccount ObjectClass:

```
dn: uid=guest2, ou=People,dc=quenya,dc=org
sambaLMPassword: 878D8014606CDA29677A44EFA1353FC7
sambaPwdMustChange: 2147483647
sambaPrimaryGroupSID: S-1-5-21-2447931902-1787058256-3961074038-513
sambaNTPassword: 552902031BEDE9EFAAD3B435B51404EE
sambaPwdLastSet: 1010179124
sambaLogonTime: 0
objectClass: sambaSamAccount
uid: guest2
sambaKickoffTime: 2147483647
sambaAcctFlags: [UX         ]
sambaLogoffTime: 2147483647
sambaSID: S-1-5-21-2447931902-1787058256-3961074038-5006
sambaPwdCanChange: 0
```

The following is an LDIF entry for using both the sambaSamAccount and posixAccount ObjectClasses:

```
dn: uid=gcarter, ou=People,dc=quenya,dc=org
sambaLogonTime: 0
displayName: Gerald Carter
sambaLMPassword: 552902031BEDE9EFAAD3B435B51404EE
sambaPrimaryGroupSID: S-1-5-21-2447931902-1787058256-3961074038-1201
objectClass: posixAccount
objectClass: sambaSamAccount
sambaAcctFlags: [UX         ]
userPassword: {crypt}BpM2ej8Rkzogo
uid: gcarter
```

```
uidNumber: 9000
cn: Gerald Carter
loginShell: /bin/bash
logoffTime: 2147483647
gidNumber: 100
sambaKickoffTime: 2147483647
sambaPwdLastSet: 1010179230
sambaSID: S-1-5-21-2447931902-1787058256-3961074038-5004
homeDirectory: /home/moria/gcarter
sambaPwdCanChange: 0
sambaPwdMustChange: 2147483647
sambaNTPassword: 878D8014606CDA29677A44EFA1353FC7
```

10.4.4.10 Password Synchronization

Samba-3 and later can update the non-Samba (LDAP) password stored with an account. When using pam_ldap, this allows changing both UNIX and Windows passwords at once.

The *ldap passwd sync* options can have the values shown in Table 10.4.

More information can be found in the `smb.conf` man page.

10.4.4.11 Using OpenLDAP Overlay for Password Syncronization

Howard Chu has written a special overlay called **smbk5pwd**. This tool modifies the `SambaNTPassword`, `SambaLMPassword` and `Heimdal` hashes in an OpenLDAP entry when an LDAP_EXOP_X_MODIFY_PASSWD operation is performed.

The overlay is shipped with OpenLDAP-2.3 and can be found in the `contrib/slapd-modules/smbk5pwd` subdirectory. This module can also be used with OpenLDAP-2.2.

10.4.5 MySQL

Every so often someone comes along with what seems (to them) like a great new idea. Storing user accounts in an SQL backend is one of them. Those who want to do this are in the best position to know what the specific benefits are to them. This may sound like a cop-out, but in truth we cannot document every little detail of why certain things of marginal utility to the bulk of Samba users might make sense to the rest. In any case, the following instructions should help the determined SQL user to implement a working system. These account storage methods are not actively maintained by the Samba Team.

10.4.5.1 Creating the Database

You can set up your own table and specify the field names to pdb_mysql (see Table 10.6 for the column names) or use the default table. The file `examples/pdb/mysql/mysql.dump` contains the correct queries to create the required tables. Use the command:

```
root# mysql -uusername -hhostname -ppassword \
   databasename < /path/to/samba/examples/pdb/mysql/mysql.dump
```

10.4.5.2 Configuring

This plug-in lacks some good documentation, but here is some brief information. Add the following to the *passdb backend* variable in your `smb.conf`:

```
        passdb backend = [other-plugins] mysql:identifier [other- ↩
            plugins]
```

The identifier can be any string you like, as long as it does not collide with the identifiers of other plugins or other instances of pdb_mysql. If you specify multiple pdb_mysql.so entries in *passdb backend*, you also need to use different identifiers.

Additional options can be given through the `smb.conf` file in the *[global]* section. Refer to Table 10.5.

Warning

Since the password for the MySQL user is stored in the `smb.conf` file, you should make the `smb.conf` file readable only to the user who runs Samba. This is considered a security bug and will soon be fixed.

Names of the columns are given in Table 10.6. The default column names can be found in the example table dump.

You can put a colon (:) after the name of each column, which should specify the column to update when updating the table. You can also specify nothing behind the colon, in which case the field data will not be updated. Setting a column name to ***NULL*** means the field should not be used.

Example 10.4.2 is shown in Example 10.4.2.

10.4.5.3 Using Plaintext Passwords or Encrypted Password

I strongly discourage the use of plaintext passwords; however, you can use them.

If you would like to use plaintext passwords, set ‘identifier:lanman pass column’ and ‘identifier:nt pass column’ to ‘NULL’ (without the quotes) and ‘identifier:plain pass column’ to the name of the column containing the plaintext passwords.

If you use encrypted passwords, set the ’identifier:plain pass column’ to ’NULL’ (without the quotes). This is the default.

Example 10.4.2 Example Configuration for the MySQL passdb Backend

```
[global]
        passdb backend = mysql:foo
        foo:mysql user = samba
        foo:mysql password = abmas
        foo:mysql database = samba
 # domain name is static and can't be changed
        foo:domain column = 'MYWORKGROUP':
 # The fullname column comes from several other columns
        foo:fullname column = CONCAT(firstname,' ',surname):
 # Samba should never write to the password columns
        foo:lanman pass column = lm_pass:
        foo:nt pass column = nt_pass:
 # The unknown 3 column is not stored
        foo:unknown 3 column = NULL
```

10.4.5.4 Getting Non-Column Data from the Table

It is possible to have not all data in the database by making some "constant."

For example, you can set 'identifier:fullname column' to something like **CONCAT(Firstname,' ',Surname)**

Or, set 'identifier:workstations column' to: **NULL** . See the MySQL documentation for more language constructs.

10.4.6 XML

This module requires libxml2 to be installed.

The usage of pdb_xml is fairly straightforward. To export data, use:

```
$ pdbedit -e xml:filename
```

where filename is the name of the file to put the data in.

To import data, use: `$ pdbedit -i xml:filename`

10.5 Common Errors

10.5.1 Users Cannot Logon

"I've installed Samba, but now I can't log on with my UNIX account!"

Make sure your user has been added to the current Samba *passdb backend*. Read the Section 10.3 for details.

10.5.2 Users Being Added to the Wrong Backend Database

A few complaints have been received from users who just moved to Samba-3. The following `smb.conf` file entries were causing problems: new accounts were being added to the old smbpasswd file, not to the tdbsam passdb.tdb file:

```
[global]
 ...     passdb backend = smbpasswd, tdbsam
 ...
```

Samba will add new accounts to the first entry in the *passdb backend* parameter entry. If you want to update to the tdbsam, then change the entry to:

```
[globals] ...    passdb backend = tdbsam, smbpasswd
 ...
```

10.5.3 Configuration of auth methods

When explicitly setting an *auth methods* parameter, `guest` must be specified as the first entry on the line — for example, *auth methods* = guest sam.

Table 10.2 Attributes in the sambaSamAccount ObjectClass (LDAP), Part A

`sambaLMPassword`	The LanMan password 16-byte hash stored as a character representation of a hexadecimal string.
`sambaNTPassword`	The NT password 16-byte hash stored as a character representation of a hexadecimal string.
`sambaPwdLastSet`	The integer time in seconds since 1970 when the `sambaLMPassword` and `sambaNTPassword` attributes were last set.
`sambaAcctFlags`	String of 11 characters surrounded by square brackets [ ] representing account flags such as U (user), W (workstation), X (no password expiration), I (domain trust account), H (home dir required), S (server trust account), and D (disabled).
`sambaLogonTime`	Integer value currently unused.
`sambaLogoffTime`	Integer value currently unused.
`sambaKickoffTime`	Specifies the time (UNIX time format) when the user will be locked down and cannot login any longer. If this attribute is omitted, then the account will never expire. Using this attribute together with shadowExpire of the shadowAccount ObjectClass will enable accounts to expire completely on an exact date.
`sambaPwdCanChange`	Specifies the time (UNIX time format) after which the user is allowed to change his password. If this attribute is not set, the user will be free to change his password whenever he wants.
`sambaPwdMustChange`	Specifies the time (UNIX time format) when the user is forced to change his password. If this value is set to 0, the user will have to change his password at first login. If this attribute is not set, then the password will never expire.
`sambaHomeDrive`	Specifies the drive letter to which to map the UNC path specified by sambaHomePath. The drive letter must be specified in the form “X:” where X is the letter of the drive to map. Refer to the “logon drive” parameter in the smb.conf(5) man page for more information.
`sambaLogonScript`	The sambaLogonScript property specifies the path of the user’s logon script, .CMD, .EXE, or .BAT file. The string can be null. The path is relative to the netlogon share. Refer to the *logon script* parameter in the `smb.conf` man page for more information.
`sambaProfilePath`	Specifies a path to the user’s profile. This value can be a null string, a local absolute path, or a UNC path. Refer to the *logon path* parameter in the `smb.conf` man page for more information.
`sambaHomePath`	The sambaHomePath property specifies the path of the home directory for the user. The string can be null. If sambaHomeDrive is set and specifies a drive letter, sambaHomePath should be a UNC path. The path must be a network UNC path of the form `\\server\share\directory`. This value can be a null string. Refer to the **logon home** parameter in the `smb.conf` man page for more information.

Table 10.3 Attributes in the sambaSamAccount ObjectClass (LDAP), Part B

`sambaUserWorkstations`	Here you can give a comma-separated list of machines on which the user is allowed to login. You may observe problems when you try to connect to a Samba domain member. Because domain members are not in this list, the domain controllers will reject them. Where this attribute is omitted, the default implies no restrictions.
`sambaSID`	The security identifier(SID) of the user. The Windows equivalent of UNIX UIDs.
`sambaPrimaryGroupSID`	The security identifier (SID) of the primary group of the user.
`sambaDomainName`	Domain the user is part of.

Table 10.4 Possible ldap passwd sync Values

Value	**Description**
yes	When the user changes his password, update `SambaNTPassword`, `SambaLMPassword`, and the `password` fields.
no	Only update `SambaNTPassword` and `SambaLMPassword`.
only	Only update the LDAP password and let the LDAP server worry about the other fields. This option is only available on some LDAP servers and only when the LDAP server supports LDAP_EXOP_X_MODIFY_PASSWD.

Table 10.5 Basic smb.conf Options for MySQL passdb Backend

Field	**Contents**
mysql host	Host name, defaults to 'localhost'
mysql password	
mysql user	Defaults to 'samba'
mysql database	Defaults to 'samba'
mysql port	Defaults to 3306
table	Name of the table containing the users

Table 10.6 MySQL field names for MySQL passdb backend

Field	**Type**	**Contents**
logon time column	int(9)	UNIX timestamp of last logon of user
logoff time column	int(9)	UNIX timestamp of last logoff of user
kickoff time column	int(9)	UNIX timestamp of moment user should be kicked off workstation (not enforced)
pass last set time column	int(9)	UNIX timestamp of moment password was last set
pass can change time column	int(9)	UNIX timestamp of moment from which password can be changed
pass must change time column	int(9)	UNIX timestamp of moment on which password must be changed
username column	varchar(255)	UNIX username
domain column	varchar(255)	NT domain user belongs to
nt username column	varchar(255)	NT username
fullname column	varchar(255)	Full name of user
home dir column	varchar(255)	UNIX homedir path (equivalent of the *logon home* parameter.
dir drive column	varchar(2)	Directory drive path (e.g., "H:")
logon script column	varchar(255)	Batch file to run on client side when logging on
profile path column	varchar(255)	Path of profile
acct desc column	varchar(255)	Some ASCII NT user data
workstations column	varchar(255)	Workstations user can logon to (or NULL for all)
unknown string column	varchar(255)	Unknown string
munged dial column	varchar(255)	Unknown
user sid column	varchar(255)	NT user SID
group sid column	varchar(255)	NT group SID
lanman pass column	varchar(255)	Encrypted lanman password
nt pass column	varchar(255)	Encrypted nt passwd
plain pass column	varchar(255)	Plaintext password
acct ctrl column	int(9)	NT user data
unknown 3 column	int(9)	Unknown
logon divs column	int(9)	Unknown
hours len column	int(9)	Unknown
bad password count column	int(5)	Number of failed password tries before disabling an account
logon count column	int(5)	Number of logon attempts
unknown 6 column	int(9)	Unknown