Sample Chapters

Windows Server 2003 Active Directory
Design and Implementation

‘The Importance of a Domain’

For more information: www.PacktPub.com/book/active_directory
The Importance of a Domain

I would assume since you're reading this book you already know you need a domain and are looking for advice and guidance on the best configuration for your environment. (If you have not bought the book and are just reading at the book shop you're taking food from my child's mouth and should stop now).

Usually people don't realize the full potential of a domain and exactly what it can offer their organization. Many times the Active Directory is simply used as a replacement for an NT 4 SAM domain without offering any additional benefits but we will discuss that in much more detail throughout this book.

Let us begin with a brief history of how the concept of domains originated and how its use has evolved up to the current state. At this point, we will investigate the current levels of functionality available. It is vital to understand where domains came from to appreciate the evolution that has occurred and the factors to consider when implementing Active Directory, which is the final destination of our adventure.

In the Beginning

The first version of Windows released in 1985 featured an amazing 256-color display and the ability to maximize windows but not a great deal more. As the versions progressed, support for memory larger than 640KB (Windows 2.x), a neater 3D interface (Windows 3.x), and an enhanced shell (the program manager) were introduced. However up to version 3.1 Windows was still nothing more than an application sitting on top of DOS with no concept of users or a network.

A network component for MS-DOS was available but this used up a large amount of memory (640KB of RAM) making it an unattractive option in addition to running Windows. To share information, users were forced to use disk media and the concept of security of the data on the removable media meant not letting it out of your sight.

Windows for Workgroups introduced built-in network support specifically around the NetBEUI protocol but also had an optional TCP/IP suite available (which interestingly can still be downloaded from Microsoft although I doubt you would receive much
support). The IPX/SPX protocol used by Novell NetWare was also supported to allow connectivity to the then dominant Network Operating System (NOS).

With this built-in network support peer-to-peer networking was possible which allowed the sharing of files and printers over the network. This sharing was not very granular; access could be read-only, or full control with some password permissioning but it was very limited. This method of resource sharing required the user to browse the network and see all machines that were network enabled and a list of shares on each visible machine. In a larger network this browsing became very cumbersome and time consuming; a method was needed to group the machines into logical or business units, hence the advent of the workgroup.

No permissions were needed to join a workgroup; you just set your machine to be part of a workgroup, e.g. sales. If you somehow misspelled the name of the workgroup (which with a name like sales would not be that easy) you would have created a brand new workgroup, and that would be your browsing start point; you would see all the machines in your new workgroup: salad (it's the closest name to sales I could think of!). When you browsed in a workgroup, you would initially see the machines in your workgroup, cutting down the number of machines that were visible. However, it was still possible to browse outside your workgroup by selecting the relevant workgroup name.

In this figure, under Windows for Workgroups, you can see the separate workgroups. In this case, the workgroups are actually domains. However, the domains also provide a workgroup-compatible interface. The Windows 98 machine is in a workgroup of the
same name as one of the domains. This is a useful technique to maintain a simpler view for machines not actually in the domain.

The workgroup concept continued to evolve, including a full account database under the NT suite of products on each of the computers called the Security Access Manager database or the SAM database, allowing accounts to be managed on a per-machine basis and allowing groups of users to be created, easing resource authorization management.

There was, however, no central user database with a workgroup; each machine held its own user and password database. This meant if you had four machines with four users, all the users would have to be created on all four machines and their passwords manually synchronized. Every time a new user was added, the addition had to be performed on all machines, and if groups were used, the group membership had to be maintained on each machine separately.

This figure is an example of workgroup configuration showing the separate user databases stored on each workgroup member machine. Because MachineC has a different password for user Hal, this may cause problems in accessing data depending on how the access has been defined.

The multiple user account databases are the primary weakness of a workgroup, which is not practical for anything over 10 machines and requires another solution.

As mentioned earlier, Novel had successful NOS called NetWare with a centralized account database system, which overcame the limitations of the workgroup model. Microsoft needed to counter this, and in collaboration with 3COM released LAN Manager (based on an even earlier NOS MS-NET, which was not very good). Originally, LAN Manager could not offer the same level of performance as NetWare but it was improving and had introduced the concept of a domain, which reworked the whole concept of user/group databases.
Newer versions of LAN Manager were released up to version 3.1 (at which point it was renamed to Windows NT) but all maintained the core concept of a domain. This domain concept remained all the way up to Windows NT 4 Server, which will be discussed here.

It is important to remember that although workgroups have now been removed from the Windows product line, even in Windows 2003 it is possible to run the machines in a workgroup configuration. For a small number of users this is simpler than the infrastructure required to run a domain. However, in most situations a domain is best suited, as we will see.

**Who's SAM?**

We stated in the last section that domains were introduced with LAN Manager, but what exactly is a domain? In its simplest form, you can think of a domain as a set of computers that share a common authentication (account) database that facilitates simpler and more secure communication between them.

This is a very different concept from that of a workgroup where each computer had its own user database. There is now one central account database used by all the machines in the domain for authentication purposes (although all machines other than domain controllers have a local authentication database as well, which is generally unused since when they log on they select to authenticate against the domains database).

As shown in the figure above, with a domain, notice there is one account database held on a central server and all the machines that are members of the domain will authenticate against that server. This database is known as the Security Accounts Manager database or SAM for short. This SAM format database was utilized by domains up to and including Windows NT 4 and is still used as the local account database for non-domain controllers. As we will see, with Windows 2000 domain controllers a brand new format was created.
Within the SAM database, each user will have one account that can be used to log on to any machine that is part of that domain; there is no need to maintain multiple accounts on each machine. The domain also provides a centralized point for network administration; all management of the accounts and other domain-related information can be performed on the server holding the domains SAM database (or any machine that has the administration tools installed with sufficient permission to connect to the domain controller). Finally, because all the computers share a single accounts database granting access to resources is far simpler.

I have referred to the SAM account database as the information replicated between the Primary Domain Controller (PDC) and the Backup Domain Controller (BDC). Actually, there is a second database replicated, the Local Security Authority (LSA) database, containing the secrets used for domain controller computer account passwords, account policy settings and trust relationships. From a practical point of view, it is not necessary to concern ourselves with the fact there are two databases; the PDC replicates its database to the BDCs. In essence, each BDC has a copy of the PDCs database, which we will discuss more in detail later.

Domain Controllers

In the previous figure a single server, the PDC, contains the account database holding all the information about the accounts in the domain and this server is known as a domain controller. In this figure, the server is actually labeled PDC, which stands for Primary Domain Controller.

Domains before the release of the Active Directory used a single-master model where only one server held a writable copy of the account database. However, only a single copy of the database is very poor from a fault tolerance and load balancing perspective. Backups alone cannot resolve this since most backups (depending on the backup schedule) are typically taken at 24-hour intervals which would mean up to 24 hours of changes could be lost in the event of a server failure and the amount of down time that would be caused by having to build a new server and restore the backup.

To counter this problem there are actually two types of domain controllers in a domain:

- **Primary Domain Controller (PDC):** The PDC holds the writable copy of the domain's account database. All modifications to domain information are performed by the Primary Domain Controller, which updates the database. There can only be one PDC in each domain.

- **Backup Domain Controller (BDC):** The BDC holds a read-only copy of the domain's account database. A BDC can authenticate user logons providing local balancing and in the event of a PDC failure can be manually promoted to the PDC role. There can be multiple BDCs in each domain.
These domain controller roles are set at installation of the operating system, and it is not possible to convert a normal server to a domain controller using the standard functionality provided with Windows NT (although several third-party vendors wrote some tools that could change the role of a server with mixed levels of success). During installation of Windows NT Server, the role of the server can be a Primary Domain Controller, a Backup Domain Controller, or Stand Alone.

As stated, in the event of the Primary Domain Controller being unavailable (if it has crashed and is not available for an unacceptable amount of time) a Backup Domain Controller can be promoted to the PDC role. The best practice is, if possible, to promote a BDC to the PDC role while the PDC is still available; this causes an up-to-date copy of the SAM to be copied to the BDC and the current PDC demoted to a BDC role.

If the PDC is in an unstartable state when a BDC is promoted, and is therefore unavailable, the PDC will still think it's the PDC. When it eventually restarts, it will detect a PDC already running for the domain and stop its NETLOGON service to avoid any possibility of corruption or lost data. The Administrator would then manually demote the old PDC to a BDC role.

The Backup Domain Controllers update their databases periodically after being notified by the Primary Domain Controller of changes. By default, the PDC would check for changes every five minutes and notify up to ten BDCs at a time (although these numbers could be modified via the registry). The notified BDCs would then wait a random amount of time before contacting the PDC and asking for replication. Using this method keeps all the databases synchronized.

There are various types of replication, full, partial and urgent/immediate. A full replication is used when a new BDC is added and when the number of changes since the last replication is greater than the size of the PDC's change log file, %systemroot%\Netlogon.chg. By default, this file is configured to a maximum size of 65,536 bytes, which normally holds 2000 changes although this can be changed via a registry change. Once the file reaches the maximum size it starts overwriting the oldest entry.

A partial replication just replicates changes since the last replication and urgent replication occurs when any of the following occur:

- An account is locked out
- A modification is made to the account lockout or domain password policy
- A machine account password changes
- A modification is made to an LSA secret

Administrators can also force a replication using the various tools available to them such as Server Manager, "net accounts /sync", and nltest, which is a resource kit utility.
The Importance of a Domain

The replication was at an object level. This means that if any attribute of an object was changed the whole object, and not just the change resulting in higher network usage, was replicated.

Joining a Domain

In a workgroup, machines were able to make themselves members by setting their workgroup value name; there was no central control or a selection committee on who could join. This is very different from a domain. Since you now have a central administration point and database, you have to be granted permission to join the domain because not everyone can be in a domain.

Unlike a workgroup, a domain is considered a corporate concept and so the "home user" versions of Windows do not support the ability to join a domain. They may access resources in a domain but are not considered part of the domain. (In fact if your workgroup account has the same name and password as a domain account then you can access resources in the domain without having to manually supply credentials!)

The table below shows the common operation systems and their domain compatibility:

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Domain Compatible?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows 95</td>
<td>No</td>
</tr>
<tr>
<td>Windows 98/98se</td>
<td>No</td>
</tr>
<tr>
<td>Windows Me</td>
<td>No</td>
</tr>
<tr>
<td>Windows NT 4 Workstation</td>
<td>Yes</td>
</tr>
<tr>
<td>Windows NT 4 Server</td>
<td>Yes</td>
</tr>
<tr>
<td>Windows 2000 Professional</td>
<td>Yes</td>
</tr>
<tr>
<td>Windows 2000 Server (all versions)</td>
<td>Yes</td>
</tr>
<tr>
<td>Windows XP Home Edition</td>
<td>No</td>
</tr>
<tr>
<td>Windows XP Professional</td>
<td>Yes</td>
</tr>
<tr>
<td>Windows 2003 Server (all versions)</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Notice that only the NT-based operating systems can operate in a domain (except for XP Home Edition). It is not just the workstation brands of Windows but also the server versions, which can operate as members of a domain. They do not have to be domain controllers to be in a domain, they can also take advantage of the central account database and are known as "member servers".
Once your client operating system is capable of being in a domain it has to be joined to the domain by an Administrator of the domain (an Administrator is like a super-user with the ability to modify the accounts database). Normal domain users cannot add computers (although this changes with the Active Directory). The computer actually has an account in the domain, just like a user, and this account can be created in advance by joining the domain via the Server Manager application or by specifying an Administrator's credentials when performing the domain-joining action, which results in the computer's account being created on demand.

The exact method of joining a domain varies slightly between the operating systems (and these are discussed later in Chapter 2) but the result will be a notification of the successful join and a prompt to restart your computer.

Once a computer is a member of the domain upon startup the user will be prompted to enter the secure-attention sequence (or Ctrl+Alt+Del as it is commonly known) which then allows the account and password to be specified.

In the logon screen shown, we see more than just one domain listed as an option to log on to. This is because of various trust relationships in place and an option to log on using the local SAM database, which we can use if we do not wish to use a domain account. Of course, in any corporate environment, users would not have any local accounts and would have to use the domain options.
The Importance of a Domain

Notice the format of the domain names, CHILD1, CHILD2, and SAVILLTECH. With the domain implementations prior to the Active Directory all domain names were NetBIOS names having a maximum length of 16 characters. NetBIOS stands for Network Basic Input/Output System, which separates the details of the network from an application by enabling the application to specify a destination for a request. NetBIOS is network independent and while originally running over NetBEUI, it was modified to also run over TCP/IP.

Since NetBIOS names can be up to 16 characters the maximum length for a domain name is actually 15 characters as the final character is used to specify the type of resource; for example <1C> is used to specify that the resource is a domain controller. A full list of the NetBIOS suffixes can be found in Knowledge Base article Q163409 that can be accessed via [link].

When you create a domain during the installation of Windows NT Server, you must enter a domain name of 15 characters or less and while some other characters are allowed you should stick to using characters A-Z, 1-9, and the hyphen character. Other legal characters are ! @ # $ % ^ & ( ) - _ ' { } . ~ although these can cause complications.

We know the domain controllers have a NetBIOS resource entry of type 1C but how will the clients actually find the domain controllers? There are three methods. The order in which they are used depends on the configuration of the client, and the options enabled on your network and clients:

- **WINS Request**: If WINS is enabled on the network when servers and clients startup they register their NetBIOS name to IP address mappings dynamically. When a client needs to resolve a NetBIOS name, such as a domain name, it sends a request to the WINS server, which will send back a list of up to 25 matching entries. WINS is mandatory in any medium-size company.

- **Broadcast**: With broadcast the client will just send out a request to its local subnet asking if anyone owns the destination name. Due to the amount of traffic created by the broadcasts and the fact that NetBIOS broadcasts are not routable, this method is only useful for small non-routed networks.

- **LMHOSTS Entry**: Each computer can have a lmhosts file, which resides in the %systemroot%\system32\drivers\etc folder (%systemroot% is an environment variable that points to the root of your Windows installation, for example, C:\windows). This file can have NetBIOS entries and one type can be for domain controllers. For example, 10.0.0.1 omega #PRE #DOM:savilltech #savilltech domain controller. This sets up IP address 10.0.0.1 to be host Omega, which is the domain controller for savilltech and instructs the machine that this entry is to be preloaded into the cache, where it would be used before any WINS lookup or broadcast.
The actual order in which a WINS request or broadcast occurs depends on the configuration node type of the client and this will be explored further in future chapters. For now, we just need to understand that the methods of finding a domain controller vary but are all based around NetBIOS domain names.

What do I Need the Active Directory For?

So far, everything we have discussed has been around the domain model used for Windows NT 4 and it was a massive improvement over the workgroup model: but it still had some significant limitations of its own, which had to be addressed:

- **40MB maximum practical database size**: 40MB may sound a lot if each account takes up around 1KB and a computer account is around 0.5KB. By the time you add groups you are looking at around 25,000 users per domain. It should be noted that 40MB was the Microsoft-supported maximum size; in reality, larger databases were possible depending on the specifications of the domain controller.

- **Replication limitations**: Very little control was given over the replication of database between the PDC and the BDCs. Some registry modifications could be made to change the period between pulses (how often BDCs are notified of changes), the number of BDCs notified at a time and other settings but these were very generic and domain controllers that were over slow WAN links could run into a lot of problems.

- **No way to delegate control**: There were effectively domain administrators (who could do everything) and everyone else (who could change nothing in the domain database). If you had helpdesk staff, who needed to reset user passwords you would have to make them domain administrators giving them a lot of power and exposing your domain to unnecessary risk.

- **No concept of physical location**: Domain Controllers and clients have no idea of where they reside physically and so clients could authenticate against any domain controller (although some workaround is possible by the use of the LMHOSTS file and the #PRE #DOM qualifiers to force remote clients to use a BDC at their location).

- **Static database format**: The SAM contained set fields: username, full name, but not much else. There was no way to add extra information and although Microsoft touted domains as a directory service this was purely for marketing reasons, to try to compete with NetWare which actually had a real directory service. If you wanted extra information you would need a separate directory/database.
There were other problems but these were the main factors in the requirement to create multiple domains (for example, if you needed more than 25,000 users or required a separate group to have control over their resources).

Once you have more than one domain you have multiple account databases. How do you enable users defined in one account database to have access to resources that are stored on computers that belong to a different domain? You just have to trust them.

**Trust Relationships**

We defined a domain as a set of machines that share a common account database and because of this common database, a domain also acts as a security boundary. Any machine not in the domain does not use our domains account database and thus cannot be granted access to our resources. Likewise, accounts in our domain cannot be granted access to resources in other domains.

Due to some of the limitations discussed above, multiple domains were a necessity in many corporate environments, but since the domains were still within one organization, a method was required to allow users in one domain to be granted access to resources in another domain. The domain that held the resources had to trust the domain containing the accounts (i.e. accept that the users had been properly authorized and were indeed entitled to an account).

This is exactly how inter-domain authorization was technically implemented: a trust relationship was created where one domain would trust another domain; the domain holding the resources would be the trusting domain (it is trusting the domain with the accounts) while the domain holding the accounts would be the trusted domain. This trust was an administrator-created communication channel to allow inter-domain authorization.

If both the domains involved need to trust each other (allowing users from either domain to be assigned access to resources in either domain), then a bi-directional trust would be created, which is actually two unidirectional trusts.

To create a trust an Administrator in the trusted domain would create a trust relationship with a password and then an Administrator in the trusting domain would complete the relationship by specifying the password set by the trusted domain Administrator. This ensured trust relationships were managed and could not be created without input from Administrators from both domains.

Another side effect of the trust relationship is any user who resides in a domain that is trusted by another domain would be able to sit down at a workstation that belongs to the trusting domain and log on to their local domain. At the main logon screen, the domain list would include the domain of which the workstation is a member and all domains its local domain trusts. As in the example shown in the figure that follows, if a user sits
down at a workstation belonging to Domain A, the drop-down domain list would show Domain A and Domain B (although workstations that belong to Domain B would not list Domain A as a drop-down option since Domain A does not trust Domain B).

In this example, Domain A trusts Domain B so accounts in Domain B can be granted access to resources in Domain A.

One final important point about trust relationships is the relationship is not transitive, this means that if Domain A trusts Domain B and Domain B trusts Domain C, Domain A does not automatically trust Domain C; this relationship would have to be manually created. This is shown in the following figure.

You have to manually create every single trust relationship between all domains that require access to resources. In large environments, this could be extremely messy especially if there was no central IT strategy.

**Domain Models**

Because of trust relationships, a number of models arose that could describe nearly all environments. These models were based around the type of trust relationships and how they were assigned. Once you analyze your NT 4 environment, it will most likely fit one of the following models.
The Importance of a Domain

Single Domain Model

In a single domain model there is a single domain containing the accounts and the resources. This is the simplest model and is suitable for most environments that are not geographically spread and have less than 25,000 users. In a single domain model, the domain administrators can administer all of the network servers.

Single Master Domain Model

A single master domain is suitable when the number of accounts required is supported by a single domain, but resource management needs to be broken up by organization. In a single master domain model, the central IT department still centrally manages the accounts in the main domain (which is the 'master domain' or the 'account domain'). Resources such as printers and file shares are located in resource domains that trust the account domain; this means the users in the account domain can be granted access to resources in the resource domains.
The multiple master domain model is very similar to the single master domain model, except there were too many accounts to be stored in one domain and so multiple account domains were required. All of the account domains have a two-way trust between them. The resource domains also trust each of the account domains.

The exact implementation varies; accounts may be split equally between domain controllers, or different account domains in different geographical regions may be present, overcoming the replication limitations of Windows NT 4 domain implementations.

One IT group can still centralize administration of the account domains or various regions can each have control of their own account domain. Like the single master domain model, the resource domain's management can be delegated to various organizational areas to give them full control of their own resources.

**Complete Trust Model**
The Importance of a Domain

No company plans for the complete trust model. Every domain has its own accounts and resources and every domain has a bi-directional trust to every other domain. This means any account in any domain can be assigned access to any resource in any domain. There is no central management or control.

Complete trust domain models typically occur as all departments have their own IT department and create their own domains for accounts and resources, but then find that access to resources in other domains is required and so trust relationships are created as required, eventually spanning all domains.

This is the hardest environment to transfer to the Active Directory due to its lack of central control. This is, however, a requirement for a successful Active Directory implementation.

The Main Event—Active Directory

We've gone into quite a bit of detail on how domains used to work before the Active Directory and this is necessary as in many situations you will be upgrading from an NT 4-based domain environment. Understanding its structure and limitations is vital when designing your new infrastructure. You need to understand the old limitations to appreciate how and why they shaped your existing structure, but these limitations no longer apply and so should not be a major factor in your new design which can be designed around business requirements and operational best practices rather than technical limitations.

Windows NT 4 SAM-based domains meet our definition of a domain: a set of machines sharing a common security database and the Security Accounts Manager (SAM) was used to manage security accounts. Other applications could sometimes leverage this authentication but it could not be used for anything else; it was not a directory service.

Microsoft realized that to compete with other Network Operating Systems it had to offer a real directory service, something that:

- Is based on the IEEE X.500 Directory Services Implementation
- Can be accessed via standard methods such as LDAP
- Can store information about all aspects of a business including applications and resources, not just users
- Can be modified to include custom attributes
- Can be fully searchable
- Allows very granular delegation of duties
- Can be scalable
One option would have been to upgrade the current domain implementation but in truth it was not a good foundation and the upgrade probably would have been more work than starting from scratch. As it turned out, Microsoft had a better starting point: Exchange.

Exchange had its own directory service for storing mailbox and distribution list information, which supported some industry standards for its interface, and so Microsoft took this directory service as a starting point.

The Directory Service Implementation

A Directory Service has to store the data and provide an interface to access the data, the same as a telephone directory service; it has a big database of all the numbers and then provides a phone number or internet page you can use to access the data.

A directory service really has to comprise three things:

- A method to store and arrange the data
- A method to locate the data
- A method to access the data

Fortunately for Microsoft, it was not inventing the concept of directory services and there were many industry standards that had been tried and tested in other implementations. By adhering to industry standards this also provided a benefit to the customer who might already have directory service tools, and allowed a far simpler migration to the Active Directory.

For the storage model of the Active Directory, the common IEEE X.500 Directory Services implementation was chosen as a standard. X.500 provides a hierarchical structure named the directory information tree (or DIT for short), which contains a number of objects, each of which comprises one or more attributes (the actual objects and applicable attributes are described in the schema, which we’ll discuss shortly). One important component of X.500 is the organizational unit that can contain other objects and even other organizational units; this is a crucial component for creating a directory service that can mimic your business model.

The following figure shows an X.500 structure with the organization broken down into three countries each having its own organizational unit structures containing objects (in this case, some users).
Each object in the DIT has two names; one is an unambiguous name, the distinguished name (DN), defining the name and exact location of the object. The other is a relative distinguished name (RDN) which only contains the name of the object relative to its position in the tree.

An example of a distinguished name would be:

CN=John Savill, OU=IT, DC=savilltech, DC=com

This shows an object by the name of John Savill in an Organizational Unit called IT in a domain called savilltech.com. Its RDN would be just "John Savill".

The actual data for the domain is now stored in a file called NTDS.DIT that is stored in the %systemroot%\NTDS folder by default. This file is based on the Microsoft Extensible Storage Engine (ESE) as used by Exchange. This new implementation throws away the old 40MB limit and a single domain can now hold millions of objects. The theoretical limit (of 4.3 billion objects) is caused by the Global Catalog and addressing issues of the i386 architecture. We will explore Global Catalog further on in the chapter.
The next issue is how to access the data in the directory. X.500 has its own Directory Access Protocol (or DAP). However, it is very large and cumbersome to implement. For this reason another IEEE industry-standard access protocol was created, LDAP, the Lightweight Directory Access Protocol, containing a subset of the full X.500 protocol.

There have been a number of versions of LDAP. However, Active Directory implements version 3 of LDAP (although it also provides backwards compatibility for version 2) and this use of an industry standard for the access mechanism allows access to the information from practically any network-enabled environment. Another advantage is that unlike X.500 (which is based around the OSI model), LDAP has full support for TCP/IP, which is obviously very useful for any kind of Internet-aware service.

LDAP operates over two main ports, port 389 for standard LDAP and port 636 for secure LDAP. Since Active Directory now provides an LDAP server, it operates over both of these ports. Anyone trying to run Exchange 5.5 on an Active Directory domain controller will face problems, since Exchange 5.5 also has an LDAP server, which tries to use port 389 and thus fails to start since Windows has already reserved it.

There are numerous interfaces to LDAP, including a programmatic API for the C programming language, because LDAP is a standard for communication between directory services, for example between a NetWare Directory Service and an Active Directory implementation. Microsoft also offers its Active Directory Services Interface (or ADSI) which offers a simple interface in communication with the Active Directory.

Microsoft went beyond the core standard version 3 of LDAP and included support for:

- Dynamic Store entries, which basically allow entries in the directory to have Time To Live (TTL) values so they can be automatically deleted (RFC 2589)
- Transport Layer Security (TLS) connection support over LDAP (RFC 2830)
- Digest Authentication (RFC 2829), which allows connection to the Active Directory to be authenticated using the DIGEST-MD5 Simple Authentication and Security Layer (SASL) authentication mechanism
- Virtual List Views (VLV), which allow clients to pull down a subset or window of results when the total result set is too large for the client to handle
- Support for InetOrgPerson class (RFC 2798); passwords can also be set on InetOrgPerson objects under Windows 2003 implementations
- Use of domains in LDAP distinguished names (RFC 2247)
- Server-side sorting of search results (RFC 2891)
- Concurrent LDAP Binds, which allow an application to bind to LDAP multiple times via one connection

While these are all functions above the core LDAP version 3, they are all still standards and should therefore be fully supported by most LDAP client implementations.
Support for the above was required to improve the whole directory service experience and provide true additional value to the enterprise.

The final problem is how to locate the services on the network offered by the directory service. The Active Directory uses Domain Name System or DNS as the location mechanism for clients to find domain controllers on the network using service (SRV) and address (A) resource records.

DNS is used to provide hostname-to-IP address mapping in a similar way to how WINS was used to map NetBIOS names to IP addresses. Unlike WINS, however, DNS is not dynamic in nature and records have to be manually created, but more on that later.

Everywhere you look DNS is in use; nearly every advertisement has a web address, for example, www.savilltech.com. This is a DNS name and is broken down into various parts but the result is an IP address with which your TCP/IP client can communicate.

The con part of the name is the top-level domain, which is a service provided by a number of Internet DNS services. The savilltech component is a second-level domain registered to a company that would be hosted by DNS servers within the company (or by the company's Internet Service Provider). The www is the actual host part of the name and would be the record looked up on the savilltech.com DNS server. This would resolve to one or more IP addresses.

This figure shows the overall structure of DNS. At the top is the root of the Internet, which is noted as a period (.). Under this are the top-level domains such as com, net, org, etc. These are all managed by the Internet root servers. Companies can then register second-level domain names, such as savilltech under a specific top-level domain to give savilltech.com. The company savilltech would have its own DNS server.
containing all records in its organization (for example a www record for its web server),
The Importance of a Domain

which in many cases would be a record known as an alias, which simply points to the actual machine offering the web service. It is not ideal to actually have a machine called www. You might want to move the service to another server or load balance it over multiple servers.

Since DNS is used as the locator service, a mechanism is needed to advertise services and the standard types of record supported by DNS were not suitable. Therefore a new type of record, the service record (SRV record), was created. The service record is now a standard for DNS and is defined in RFC 2782. However, not all DNS implementations support it and service records are a mandatory requirement for the Active Directory. Without service records, Active Directory will not function.

Once you install an Active Directory domain you will see a large number of records added to DNS, which are service records and basically provide the clients on the network with a way to find the domain controllers by searching for LDAP service records (although the actual records are of the form _ldap._tcp.<domain name>).

The lines that follow show the DNS records added when creating a domain. These values are stored in the netlogon.dns file created when a domain controller is promoted. This file is also dynamically re-created at the default refresh period (24 hours), or each time the Netlogon service is restarted, or when the domain controller is restarted. This data is dynamically (or manually) registered into DNS when using a Microsoft DNS server or a version of DNS, such as BIND, that supports SRV records. Ideally, of course, the idea is to use a server that supports dynamic updates. It is better than attempting to manually create all these records in DNS:

```
savilltech.com. 600 IN A 10.0.0.1
gc._msdcs.savilltech.com. 600 IN A 10.0.0.1
DomainDnsZones.savilltech.com. 600 IN A 10.0.0.1
ForestDnsZones.savilltech.com. 600 IN A 10.0.0.1
_ldap._tcp.savilltech.com. 600 IN SRV 0 100 389 omega.savilltech.com.
_ldap._tcp.pdc._msdcs.savilltech.com. 600 IN SRV 0 100 389 omega.savilltech.com.
_ldap._tcp.gc._msdcs.savilltech.com. 600 IN SRV 0 100 3268 omega.savilltech.com.
_ldap._tcp.1e95687b-3e01-44f9-adb3-70e1602237e3.domains._msdcs.savilltech.com. 600 IN SRV 0 100 389 omega.savilltech.com.
1ea880a3-5df2-44b2-8229-a1cbf3d3709._msdcs.savilltech.com. 600 IN CNAME omega.savilltech.com.
_kerberos._tcp.dc._msdcs.savilltech.com. 600 IN SRV 0 100 88 omega.savilltech.com.
_ldap._tcp.dc._msdcs.savilltech.com. 600 IN SRV 0 100 389 omega.savilltech.com.
```
_ldap._tcp.Gotham._sites.dc._msdcs.savilltech.com. 600 IN SRV 0 100
We discussed that DNS was static and would have to be manually updated but after installing Active Directory, you will see a large number of additions. This would seem to be a contradiction. The next requirement of DNS by the Active Directory is **dynamic update**.

There was a whole page of records added for our domain; manually creating all of these would be a daunting task and for the most part every domain controller would have that amount of records again. In order to aid in the management and cut down on the number of human errors, DNS has been improved to allow DNS clients to register their own records. In this case, domain controllers should be allowed to register all the records (including service records) required for clients to "find" the offered services on the network. Dynamic DNS is another standard and is defined in RFC 2136.

Unlike mandatory service records, dynamic update is not a "must-have" but rather a "really useful to have and mandatory unless you have a brain the size of a planet". It is
possible to manually create all of the records required; however, it is highly
recommended to use a Windows 2003 DNS server for the DNS namespaces related to the Active Directory.

**The Blueprint of the Active Directory**

Active Directory needs to have far greater scope of attributes than available under NT 4 domains because it is a directory service designed to hold information about all parts of your organization. However, it is impossible to create all possible required fields; what is needed is some kind of plan for the information that can exist in the Active Directory implementation that can be modified if needed.

What the Active Directory schema provides is a definition of the objects (or classes) available and the attributes the objects have. These attributes are defined separately from the objects and then linked to the object definitions, allowing objects to be defined using any attribute described in the schema.

What is vital with the schema is it can be dynamically extended, defining new classes and attributes or adding new attributes to existing classes. Modifying the schema is not to be taken lightly and has effects on replication throughout the entire organization. Nevertheless, it is a very useful ability.

Only one domain controller in your entire organization (assuming you have one 'forest') can modify the schema. The user trying to modify the schema will need to have certain permissions; you should restrict the normal administrators’ ability to change the schema; only very high-level administrators should have schema modification abilities.

Once you start installing newer backoffice applications you will start to understand how vital the schema is and how the Active Directory is the foundation of your entire infrastructure. Nearly every single major backoffice service wants to extend the schema (some more than others!). Exchange, Systems Management Server, and SharePoint are just a few of the services that extend the Active Directory schema with new attributes and classes.

**Creating a Domain Controller**

During the installation of Windows NT 4 Server, you were asked if the server would be a PDC, BDC, or standalone server. When you install Windows 2000/2003 Server you are no longer asked this question, so how do you create a domain controller?

This was always a major problem with NT 4 servers. Defining the server's role at installation time was a pain and this has now been resolved. Rather than defining a server's roles during installation you now run a wizard after the operating system installation is complete, to change the server to a domain controller or to change a domain controller to a normal server.
This wizard is known as DCPROMO (Domain Controller Promotion) and has a step-by-step process to guide the whole process (an example is provided in Chapter 2).
You can now technically switch a server's domain controller status (that is not to say you should perform this without planning; there are still other factors!).

One thing you will notice even with the wizard is the lack of a PDC/BDC option. Surely, you can have more than one domain controller per domain with the Active Directory.

**Domain Controller Farm**

There have been huge advances with the Active Directory: a single domain can hold millions of objects and the performance of domain controllers is much higher thanks to more efficient authentication protocols, but you will still want more than one domain controller for fault tolerance and load balancing reasons.

The concept of a single master domain controller has gone with the Active Directory; now multi-master replication is used, which means any domain controller can make changes to its copy of the domain information and through a process known as multi-master replication the databases on each individual domain controller are kept synchronized.

It would seem that all domain controllers are equal but some are more equal than others. Some functions do not lend themselves to multiple masters. We will talk more about these functions once we have looked at more of the domain concepts.

The replication topology varies depending on the location of the domain controllers. Replication is more frequent if the domain controllers are in the same physical location and can be highly tailored between physical locations. We will cover this in more detail later in the book.

One important factor to remember is that Windows NT 4 BDCs are still supported in Active Directory domains (but not an NT 4 PDC; if you upgrade, it has to be PDC first) and these BDCs do not support multi-master replication: they only pull updates from one domain controller—the PDC. Since this PDC no longer exists, a single Active Directory domain controller in each domain pretends it is an NT 4 PDC for the benefit of the NT 4 BDCs.

**Kerberos**

NTLM is an old protocol and while it still works and is effective, it has some limitations:

- It's not a very fast protocol and has quite a high overhead.
- Each client access requires the server to contact a domain controller for verification, putting load on the server.
- There is a proprietary protocol cutting down on supportability.
- No support for delegation of authentication is provided.
• Servers are not able to authenticate with other servers.
The Importance of a Domain

For the Active Directory, Microsoft has chosen Kerberos as its default protocol (although NTLM is still supported for backwards compatibility). Kerberos is an industry standard defined in RFC 1510 and takes its name from the three-headed dog that guarded the gates of the underworld in Greco-Roman mythology.

The three-headed part comes from the way in which Kerberos works: the client, the server the client wishes to use, and the trusted third party, which is the Key Distribution Center providing authentication. It's important to understand that it's not only users who contact the KDC for access to a server, services on servers also contact the KDC to enable access to other servers.

This figure illustrates the major steps involved in a client communicating with the KDC to get the information needed to talk to a server. The idea is: if two people know a secret they can communicate and if only they both know the secret they know the other person is who they say they are. You cannot just send the secret over the network as plain text because anyone with a network sniffer could find the "secret".
The Kerberos protocol solves this problem with secret key cryptography. Rather than sharing a password, communication partners share a cryptographic key—symmetric in nature—that can both encrypt and decrypt.
The Importance of a Domain

The process starts when the user first logs onto the domain. It works as follows:

- The user enters the username and password at the logon screen. The local Kerberos client converts the password to an encryption key by creating a one-way hash value.
- The local client time is then encrypted with the generated encryption key and a KRB_AS_REQ (Kerberos Authentication Service Request) is generated containing the user's name, the request for a ticket-granting ticket, and the encrypted time. This is sent to the Authentication Service component of the Key Distribution Center (KDC).
- The KDC (a domain controller having access to the Active Directory) looks up the user's information including the password hash and uses it to decrypt the time. If it is within five minutes (the time limit can be changed) of the server's time, it knows it is not a replay. Once the user is confirmed, the KDC creates a session key used for future communication between the user and the KDC. This generated session key is encrypted with the user's encryption key (the hash of the password) and also encrypted with the KDC's own long-term key, which is known as the ticket-granting ticket. This is sent back as a KRB_AS_REP (Kerberos Authentication Service Reply).
- The client now wishes to talk to a server. It sends a request to the Ticket-Granting Service component of the KDC containing the ticket-granting ticket and the server it wishes to talk to in a KRB_TGS_REQ (Kerberos Ticket-Granting Service Request).
- The KDC decrypts the KRB_TGS_REQ using the ticket-granting ticket (which it decrypts with its own secret key) and assuming it passes authentication testing, generates a session key to use for communication between the user and the desired server. This session key is encrypted with the user's encryption key and with the server's long-term key in the form of a ticket. These are then sent to the user in a KRB_TGS_REP (Kerberos Ticket-Granting Service Reply).
- Now the client can initiate communications with the server by sending a KRB_AP_REQ (Kerberos Application Request) containing the user's name and the time encrypted with the session key to be used between the user and the server along with the server's ticket (which is the session key encrypted with the server's long-term key).
- The server decrypts the ticket using its long-term key and extracts the session key. It can then decrypt the encrypted time and if it passes, the server can trust the client's identity. If the client asked for mutual authentication, the server then encrypts the time using the session key it
shares with the user and sends it back as a KRB_AP_REP (Kerberos Application Reply).
The Importance of a Domain

- The client at the user's workstation then decrypts the KRB_AP_REP if requested. If the authentication passes, then the client knows the server could decrypt and use the ticket, proving they are who they said they were.
- The client and the server now have a mutual session key that they can use to encrypt any required communication.

You will notice that at no time do any of the servers have to remember anything about the client. The client always sends the server a ticket generated by the KDC for its use with all client communication. The server never has to contact the KDC directly during client/server session initialization.

For each server the user needs to communicate with, a separate ticket will be created and the process of the KRB_TGS_REQ—KRB_TGS_REP—KRB_AP_REQ—KRB_AP_REP performed.

These issued tickets only last a certain amount of time and the KDC does not keep track or notify clients of their expiry, but in the case of an expired ticket or even an expired ticket-granting ticket, a new one will be requested using the discussed steps. The user's password hash is cached. However, if for some reason it is no longer cached, the user may be asked for their credentials again.

You will notice that in nearly all steps the time is encrypted with the session key to prove it is not just a replay. Since time is part of the encryption technology machines using Kerberos need to be time-synchronized with an SNTP service.

There are more steps involved if the server is in a separate domain and this will be covered in later chapters.

**Domains**

The concept of a domain has remained in the Active Directory and as with an NT 4 style, a domain acts as a boundary of replication for domain naming context, which contains information about objects contained in the domain, for example users, security policies etc. This is the equivalent of the replication boundary of the SAM database under NT 4-based domains.

Domain controllers in a domain only receive information about the objects located in the domain. This replication boundary is what allows the Active Directory to scale so well by allowing multiple domains but not replicating all the information throughout the entire enterprise.

Unlike Windows NT 4 domains, Active Directory domains are DNS (Domain Naming System) names, which is the format you see when using the Web. www.savilltech.com comprises a host record www in the DNS domain savilltech.com. Unfortunately, this is where some confusion can creep in, since Windows uses the term domain to define a
group of computers sharing a common security database, whereas DNS uses the term
domain to specify the name of a portion of the DNS namespace. For example, a DNS domain name could be savilltech.com but since Active Directory uses DNS for its naming mechanism, savilltech.com can now also be the name of an Active Directory domain.

NT 4 domains had 15 character NetBIOS names. It was a flat single level so there was no hierarchy in the name; now we use DNS with a hierarchical name space allowing a hierarchical view of the systems. For example, sales.uk.savilltech.com and legal.acme.com are valid Active Directory domain names because they are valid DNS domain names. This will make more sense when we talk about trees. If you upgrade an NT 4 domain to Active Directory, you cannot change the NetBIOS name during the upgrade process. If you are creating a new domain, you can name the NetBIOS name anything you want; by default, it will be the first 15 characters up to the left-most period of the DNS domain name, so if your DNS domain name was sales.savilltech.com, by default your NetBIOS name would be sales.

Because domain names are DNS names, a fully functional DNS infrastructure is mandatory; however Windows 2003 can help the configuration if necessary.

Trees

As we just saw in the previous section, Active Directory domains are now DNS names with a hierarchical structure. This allows the creation of an Active Directory domain hierarchy with a contiguous namespace, known as a tree.

The figure above shows a typical Active Directory tree. Notice the contiguous nature of the naming. The domain uk.savilltech.com has the name of its parent domain (savilltech.com) as part of its name.
This hierarchy of domains is formed at the time of domain creation. When you run the DCPROMO wizard to promote a server to a domain controller, you are given the following options:

- Make the server a new domain controller for an existing domain
- Create a new domain as a child of an existing domain (which would add it to an existing tree)
- Create a new domain in a new tree
- Create a new domain in a new forest

If you select 'Create a new child domain' you will be asked to enter the name of an existing domain and this entered domain will become the parent domain for the newly created domain.

Under Windows NT 4, manual trust relationships could be created. Under the Active Directory when a child domain is created a transitive bi-directional Kerberos trust is automatically created between the parent and the child domain. This trust does not need to be manually managed; it is a feature of the parent-child relationship.

Transitive trust is a change from NT 4 domains where, if domain A trusted domain B, and domain B trusted domain C, domain A did not automatically trust domain C. Under the Active Directory transitive Kerberos trust, this trust would automatically exist.

Going back to our domain tree example (the previous figure) notice that each domain has a bi-directional trust with its parent/child and because these are transitive, it means every domain in the tree implicitly trusts every other domain.

The relationships shown by the dashed lines in the figure above implicitly exist because of the transitive nature of the trust relationships between the parent and child domains.
The Importance of a Domain

This is great for administrators, it means now any user or group from any domain in a tree can be granted access to any resource in any domain in the tree without a single trust relationship being created.

We previously saw how domains acted as a boundary of replication, with each domain containing the information about the objects in the domain. This is known as the domain partition, which is one of the three logical partitions of the physical Active Directory database. This partition is also called the DomainNC (the Domain Name Container), which contains only that specific domain's data, such as user accounts, group accounts, computer accounts, and other objects that were created in that domain. Every domain has its own separate domain partition, and every domain controller in that specific domain has a full replica (copy) of that domain's domain partition, which is replicated among only those DCs in that specific domain.

There are two other partitions within the Active Directory that are common to all domains in the tree, i.e. they all contain the same information and are replicated to all domain controllers in a forest:

- **Schema partition**: As we previously saw, the schema is the blueprint for Active Directory and the schema partition is used to contain that blueprint definition. The fact that this schema partition is common to all domains in a tree should confirm the fact that all domains in a tree have a common schema. You cannot change the schema for just one domain in a tree; if a change is initiated anywhere in the tree it will propagate throughout the entire tree (in fact the schema can only be changed at one point in the tree which we will see soon). Every single domain in the tree has a replica of the same schema partition.

- **Configuration partition**: This contains the replication topology and other configuration information that is common to the entire tree. This includes information about the domains in the tree, domain controllers and sites. Every single domain in the tree has a replica of the same configuration partition.

Windows 2003 introduced a new directory partition type—the application directory partition. The application directory partition stores dynamic application-specific data in the Active Directory but rather than being replicated to all domain controllers in a domain or tree, the data is replicated only to domain controllers specified by the Administrator. Application directory partitions can contain any type of object apart from security principals (users, groups, and computers).

The data contained can be configured to replicate to any domain controller in any domain within the tree or every domain controller in the tree. All of the domain controllers configured to host the application directory partition hold a replica of the information. However, only Windows 2003 domain controllers can host a replica of an application
directory partition.
Windows 2003 actually uses the application partition to enable the ability to replicate DNS information stored in the Active Directory to only specific domain controllers. You also have the option with Windows 2003 to replicate DNS information to only DNS servers in the forest or domain, which is implemented via two application partitions created automatically, DomainDNSZones and ForestDNSZones. Each domain has a separate DomainDNSZones partition (in the same way each domain has its own Domain partition) that every domain controller hosting the DNS service contains a replica of. The forest has a single version of ForestDNSZones, of which every domain controller hosting the DNS servers contains a replica.

This overcame a limitation in Windows 2000 Active Directory, since these application partitions did not exist. When we created AD Integrated zones in Windows 2000, they were created in the Domain NC and therefore could not replicate to other domains. So if we needed that zone in a child domain (if not delegating to the child DNS or not using stub zones for the child DNS servers), or in another domain in the forest, we needed to create a secondary zone to make that available on the DNS server in the other domain. With Windows 2003, we can specify whether we would like this zone to replicate either domain wide or forest wide so that it will be available in the application partition so we can have that zone available in the other domain. We will explore this further in future chapters. These three core partitions are shown in the following diagram:
Each domain here has a different domain partition shared just between the domain controllers in the domain, while all domain controllers have a common schema and configuration-naming context.

To summarize, a tree is a set of domains that form a contiguous namespace with parent/child domains connected via transitive Kerberos trusts. This means every domain in the tree trusts every other domain. The domains in a tree also share a common schema definition. How, then, will you handle more than one namespace? You will need another tree.

**Forests**

A forest is one or more trees connected at the tree roots by Kerberos bi-directional transitive trusts. As with a tree, this now means that every single domain in a forest trusts every other domain in the forest and in other trees.

As you can see, the two trees in the figure are connected via a transitive Kerberos trust, meaning all domains trust each other as if they had an explicit trust (the dotted lines). With a forest, you can have multiple namespaces and still enjoy the advantages of a single Active Directory structure.

When we looked at trees, there were a number of common features between all the domains in a tree. These common features are actually common between all domains in a forest (except for the contiguous namespace). This includes a common schema between all the domains in the forest and a single everyone group as well as the authenticated users group.

A new tree is added to the forest during the DCPROMO process; you cannot add an existing tree in its own forest to an existing forest for practical reasons. If they had different schemas and all domains in a forest had to share a common schema it would
create a problem. However, Windows 2003 does introduce the ability to create a transitive trust between separate forests as long as all domains and forest are at full Windows 2003 functional level.

Organizational Units
Organizational Units (OU) are one of the greatest features of the Active Directory. They offer long-overdue functionality for Windows-based directory services and were a necessity for Microsoft in adding this functionality to organize our objects in a scalable way that is customizable and flexible for any business needs. Microsoft did a great job in adding this functionality to help simplify our design during the initial domain design phase and easing the process of altering the design well after it has been implemented, and thus increasing the ease of the network management going forward. As we will see, Organizational Units in many cases do away with the need of resource domains.

With previous domain implementations, users could be put into groups, enabling control over resource access authorization, but not much else. Groups still exist (and have been expanded) in the Active Directory for the purposes of simplified Access Control List (ACL) management but they do not help if you want to group users and resources for the purposes of management, of policy application or to actually hide objects.

Organizational Units are containers that can hold nearly any other type of objects including other Organizational Units to form a hierarchy. Organizational Units, as the name implies, are used to organize objects into logical groupings and it is important to understand that Organizational Units are an Administrators’ tool; they should be created to ease the administration of your environment and primarily used for the following reasons:

- **Delegation of Authority:** It is possible to assign people/groups with administrative permissions over Organizational Units. These permissions are far more granular than under NT 4 domains. Instead of being a full administrator, it's now possible to delegate just the ability to reset users' passwords, or modify only the telephone number attribute of users. Delegation is also possible at other levels (for example, at domain level) but an OU is the smallest scope at which delegation can occur (you cannot delegate at a group level).

- **Group Policy Application:** Group Policy has replaced the old System Policies and can be assigned at multiple levels, one of which is group level. Since Organizational Units can be nested, Group Policy could be applied at
all levels of the OU nesting giving a lot of flexibility (maybe too much) in the resultant policy applied to the computer or user.

- **Hiding Objects**: If you have Active Directory resources that should not be visible when browsing, you can place them in an Organizational Unit and then configure the OU so that certain groups of users cannot view the content.

- **Logic grouping of resources to aid administration**: It's possible to perform administrative functions on more than one object at a time; to ease this object selection you could place objects in OUs based on how they are typically administered. Care should be taken in this case and OU creation should primarily be based on the first three reasons (this reason can sometimes lead to a large number of OUs, which can eventually increase the complexity of an environment and adversely affect performance).

It is important to understand the difference in implementation of groups and OUs. When users are placed in a group, the actual user object is not moved but a reference is placed in the group to show membership; when an object is placed in an OU, it is actually moved to inside the OU, hence you cannot place an object in more than one OU. The exception is that since OUs can be nested, if an object is placed in OU B and OU B is in OU A then such an object would inherit settings (including delegation and group policy) applied to the OUs A and B.

Organizational Units are not used for ACLs; you cannot assign an OU to an ACL to allow all users in an OU access to a resource, you would still have to use a group for this purpose. OUs can have permissions assigned to them (for hiding etc.) but you cannot use them for ACL purposes.

OU structures are separate for each domain. Each domain can implement its own OU hierarchy, and it is not possible to share an OU structure between domains (even if they are in the same forest). An OU is stored in the domain partition, and is not available across domains.

In each domain one OU is created automatically: the 'Domain Controllers' OU into which all domain controllers are placed. This is because a default Group Policy Object exists for domain controllers which is applied to this default OU.

In the figure that follows, both domains contain separate OU structures, which in turn contain a mixture of resources:
The Importance of a Domain

Sites

So far, the components we have looked at have been logical components, forests, trees, domains, and OUs. We can design these items. However, there are some physical components, the physical location of your users, computers, servers, and the connectivity between these locations that you cannot 'design' in the normal sense; instead, you have to document these to the Active Directory to aid in its function.

To document your physical structure in a way that can be understood by the Active Directory, use IP subnets. Since Active Directory is based around DNS, which in turn requires TCP/IP, it is the most logical choice. IP subnets are always a local LAN area, e.g. a floor of a building. (If you have a single subnet over a wide area network you need to correct this before implementing AD.) It is very common for one location to use multiple subnets (if, for example, it has too many machines to be covered by one subnet).

The first step is to define all the IP subnets in your environment and then enter your physical locations with the IP subnets linked in those locations. The Active Directory now has the name of each site and knows what IP addresses reside within it. Since every computer has an IP address, its physical location can be determined due to the link between IP subnets and sites.

Which IP subnets should be placed in a site? As long as the subnets are geographically within the same area and are connected via a LAN, for example over 10Mbps (although a speed as low as 512Kbps can be adequate), they can be configured in a site. A site is a group of computers communicating via high-speed, reliable connectivity.
Once the links between the sites are manually defined, which includes the speeds of the links (which are defined by costs; the lower the cost, the faster the link), the availability of the links is also defined (for example, a link may not be available at night).

In this company, in the preceding figure, four geographical locations are used with the various IP subnets defined. The network speeds are also documented (although these would be converted to a cost when defined in the Active Directory). Notice that for this company Dallas is really the hub; although San Antonio and Houston do have some limited communication, it would still be faster for them to communicate via Dallas.

Defining all the sites when the connectivity between them is a lot of work initially—why does the Active Directory need to know the physical locations of the machines on the network? This is for two main reasons:

Firstly, one of the major problems with Windows NT 4 domains was replication over WAN links. Very little control was possible regarding replication; NT 4 had no way of knowing where the domain controllers physically resided or the connectivity between them, and so, very little configuration was possible.
With the sites defined, the Active Directory is aware of where the domain controllers are physically located and the network connectivity between them. This knowledge is used by the Knowledge Consistency Checker (KCC), to create replication connection objects between domain controllers to define a replication topology used for the replication of the Active Directory data.

The second main use is that since clients have IP addresses they know which physical location they reside in. If the client needs a service (say, a domain controller), it can be told to use a domain controller in its local location (or a closest location, if no domain controller is available in its local site). This is known as site awareness and most major Active Directory services are site aware, minimizing traffic sent over slower WAN-type links by sending clients to services in their local site, where possible.

By default, the KCC component runs every 15 minutes checking that the topology generated is the most efficient. The KCC runs on every domain controller, which may lead you to believe it would be possible for domain controllers to create different topologies leading to problems with the replication; but such problems do not arise. Each domain controller uses the same algorithm to create the topology, and has the same information about sites and the domain controllers within them. The same inputs to an algorithm will lead to the same result.

The actual connection objects created by the KCC vary if the connection objects are created within (intra) a site or between (inter) sites. Remember that within a site, a high bandwidth network, a LAN, connects all the computers and so network bandwidth is plentiful and replication can be based around the fastest update of all the domain controllers in the site; the traffic generated is not a concern.

To this end within a site, a ring topology replicates information. Every domain controller has at least two connections to other domain controllers within the site; sometimes more connections are created to ensure there are never more than three hops between any two domain controllers. This replication is trigger based; whenever a change is made within a defined time (five minutes by default) the domain controller with a change, will notify its replication partners and they will pull the change. No compression is used when sending the data, as the CPU time required to perform the compress/decompress would be more expensive in terms of resources than the additional network use.

Between sites, the network may not be fast and reliable and so the number of connections and replication traffic are minimized. To achieve this instead of a ring, a least-cost spanning tree is used, which ensures that all sites are connected in the cheapest way possible. Because the network is slower, replication is based around a schedule, for example, every 30 minutes replication occurs between site A and B. To save network bandwidth the information is compressed if it is over 32KB in size.
In this example, each site has its own ring of replication and then both San Antonio and Houston have a single replication connection object for the inter-site replication.

One big change from NT 4 is that for intra- and inter-site replication only the modified attribute is replicated. Previously if a single attribute changed the whole object would be replicated, but now only the modified attribute is replicated, saving a lot of bandwidth and computational processing.

Site definitions and site links are stored in the Configuration naming context; this means it is common to the entire forest. You do not have to define your sites for every domain, and you define them once for the entire enterprise so that domain controllers from different domains can reside in the same site. This is important as it's not just the domain information that has to be replicated, but also the schema, configuration, and application partitions. These also need their own replication topologies so as to replicate across domain boundaries and, like domains, different rings/spanning trees are created depending on domain controller site location. As we will see later on, there are multiple rings of replication running within a site performing different functions.

There is far more to sites and the replication used and we'll cover this in much more detail when we get to the appropriate scenario, as this is one of the key areas that, when implemented well, results in an efficient, well connected directory service environment.

**FSMO Roles**

The other physical component is the domain controllers themselves. As we have seen, in the Active Directory all the domain controllers are equal. There is no PDC or BDC; all the controllers are equal, performing multi-master replication.
The Importance of a Domain

We have also seen that NT 4 BDCs can participate in an Active Directory domain. How is this possible since NT 4 BDCs have to pull information from the PDC, which no longer exists?

Some domain controllers are more equal than others. In fact, there are five special roles that certain domain controllers hold to perform functions that cannot work in a multi-master fashion. These are known as Flexible Single Master Operation or FSMO roles. The first of these relates to handling NT 4 BDCs; that's not all it does, but it's a start.

**PDC Emulator FSMO Role**

A single domain controller in each domain holds only the PDC FSMO role. You always have one PDC FSMO role per domain in the same way that you used to have one PDC per NT 4 domain.

The PDC FSMO performs a number of functions:

- It provides the replication point for NT 4 BDC's in the domain.
- Each PDC FSMO in the forest synchronizes its time with the other PDC FSMOs within the forest. The PDC FSMO in the root domain of the forest is the source of the time synchronization and is normally configured to synchronize its time with an external time source. All clients and servers within each domain synchronize their time with their domain's PDC FSMO.
- It provides down-level clients support for password updates. Since non AD-aware clients know BDCs are for read-only, they would always attempt to change passwords with the PDC.
- It acts as the Master Domain Browser if the service is enabled.
- Password changes at any DC are always replicated to the PDC FSMO first. This is because replication can take time to propagate the password change and if a user has just changed their password and attempts to authenticate again against a different domain controller the new password may not have replicated and so the authentication attempt would fail. To avoid this, if the authentication fails at a domain controller instead of rejecting the request the domain controller will first contact the PDC FSMO role holder to attempt a second authentication before a failure is passed back to the user.
- Due the criticality of locking out an account, this is always processed at the PDC FSMO to provide a central location for checking the status of accounts.
- Where possible the PDC FSMO will be contacted for Group Policy maintenance (edit/create). However, other DCs can be used via configuration.

As seen, even when there are no NT 4 BDCs or non-AD clients the PDC FSMO role is still required though its workload would be reduced.
The normal site awareness of replication does not apply to NT 4 BDCs, which, as seen in this figure, will always replicate with the PDC FSMO role holder, regardless of which site it resides in and the cost involved. This becomes an important factor when considering the placement of your FSMO role holders.

**RID Master FSMO Role**

Every object in the domain has a Security Identifier, known as a SID. This SID is composed of the SID of the domain the object resides in and a Relative Identifier (RID), which is unique in each domain.

These Relative Identifiers have to be unique in the domain. If each domain controller in the domain made up its own, there is a chance it would clash with that created by another domain controller. So, a single domain controller in each domain, the RID FSMO, gives batches of 500 RIDs to each domain controller. When a domain controller has only 100 RIDs left (20%), it requests another batch of 500 from the RID FSMO. However, with Windows 2000 Service Pack 4 and above, a new batch is requested when 50% (or 250) of the number of RIDs are left, improving resilience if the RID FSMO is not available.

The RID Master is also used when moving an object between domains (this is now possible quite easily, thanks to some tools that are supplied with Windows: for example, the `movetree.exe` utility). Even though you may not specifically run the utility on the RID server, the utility will automatically contact and use it.

This role is also per domain and so every domain has a RID Master FSMO server.

**Infrastructure FSMO Role**

It is possible for an object in one domain to be referenced by another domain. For example, when a user from domain A is placed in a local group in domain B, the reference information stored in the domain B group is:
The Importance of a Domain

- The Globally Unique Identifier (GUID) of the object (which never changes during the objects lifetime, even if it is moved between domains)
- The Security Identifier (SID) of the object (which would change if moved between domains)
- The Distinguished Name (DN) of the object (which changes if the object is moved in any way)

This information is stored in a record known as a phantom record.

The Infrastructure FSMO is responsible for ensuring that the SIDs and DNs of the phantom records of objects referenced from other domains are kept up to date by comparing the content of its database with that of the Global Catalog. If the information it has stored for the GUID of the object is different from that of the Global Catalog, the phantom record is updated with the new information. This checking process runs periodically and it is possible that sometimes you may view a group and you will see grayed icons. This simply means the object with the DN cannot be found at present; it has probably moved and the infrastructure master has just not updated the phantom record yet. It is not a problem and would not affect the working of the group.

It is vital that the Infrastructure Master FSMO role is not a Global Catalog if you have more than one domain, since its database will never see anything different from the Global Catalog (since it is one). If you only have one domain, it does not matter since you will not have any objects from other domains referenced.

Again, this role is a one-per-domain deal.

**Schema Master FSMO Role**

We talked about the schema being the blueprint for every class and attribute within the entire forest and that it was policed so that only certain users could request a change (members of the Schema Admins group) and changes could only be requested through a particular domain controller, the Schema Master. This time, however, since the schema is forest wide only one domain controller in the entire forest has the role; only this machine has the ability to write to the schema, which is then replicated to every other domain controller in the entire forest.

**Domain Naming Master FSMO Role**

During the DCPROMO process, domains can be added to a tree or forest by selecting a parent domain. We need to ensure the same domain name is not used twice and that every domain name is unique in the forest. This is the responsibility of the Domain Naming Master FSMO, which has to be contacted before any domain can be added or removed from the forest. With Windows 2003, it is also used when moving domains around the forest structure (we can move domains around the forest, which is known as "prune and graft").
This figure shows that within the forest there is a single instance of the Schema and Domain Naming FSMOs (they happen to be on the same domain controller but this need not always be the case) and that every domain has its own PDC, RID, and Infrastructure FSMOs (again these do not have to be on the same server).

All of the domain-specific roles will be held by the first domain controller created in each domain, by default. The roles can be moved to any domain controller within the domain (but not to an NT 4 BDC) and can exist on the same server or split across multiple domain controllers. There are some best practices for their placement, which will be examined in more detail later.

The forest-specific roles will, by default, be held by the first domain controller ever created in the forest (which will be in the forest root domain). These roles can be moved too.

It is important to understand that these roles do not move automatically; if the server running one of these roles goes down, the functions it performs will not be possible: for example, if the RID FSMO is unavailable for an extended time you may no longer be able to create new objects in a domain.

**Global Catalog**

A Global Catalog server is a special domain controller that not only holds a full replica of its local domain partition but also a read-only copy of a subset of attributes of every object in every other domain. It is important to understand that Global Catalog is not an FSMO role, but rather an additional service running on specific domain controllers.
The Importance of a Domain

By default, the first domain controller created is a Global Catalog (GC). However, any domain controller can be configured as a GC and there are a number of best practices regarding this configuration, which will be examined later in the book.

The attributes stored in the Global Catalog are defined as the Partial Attribute Set (PAS) and the PAS can be modified by marking or unmarking attributes in the schema as replicated in the Global Catalog.

Each domain's domain controllers, as seen in the figure above, hold a full replica of the local domain's partition and the global catalogs also contain a subset of every other domain's partition.

The Global Catalog is used to locate resources within the enterprise. A domain contains full information about resources in its domain but trying to find resources in the rest of the forest would be very time consuming if a domain controller in each domain had to be located and queried for any search. Instead, enterprise queries are directed at a Global Catalog.

In addition to providing an enterprise search ability the Global Catalog is used to store a specific type of group, the Universal group, which is accessible from any server in the forest and can contain users for any domain in the forest. A Global Catalog is queried during logon to check for Universal Group membership. If a GC cannot be contacted, then users will not be able to logon unless a new feature of Windows 2003, the ability
for sites to cache Universal Group membership, is used (although domain administrators can log on even without GCs).

Obviously the Global Catalog content has to be replicated between every Global Catalog in the entire forest. (There is only one ‘version’ of the Global Catalog; they should all share the same content assuming that the replication was instant. Nevertheless, there will be some minor differences due to replication latency.) The Knowledge Consistency Checker (KCC) creates additional connection objects for this GC replication, and connection objects to domain controllers in other domains, to ensure that a subset of every domain’s partition content is available in the GC.

When a user provides a query to the Global Catalog, they first ask the DNS server for a Global Catalog. Once a Global Catalog is returned, they perform the query via port 3268 on the server (normally port 389 is used for a standard LDAP query). If the Global Catalog does not have the attribute being queried as part of the PAS, the query is referred to the normal LDAP Active Directory service.

Global Catalog servers will not return any data stored in an application directory partition that they hold a replica of: only information originating from domain partitions is returned via Global Catalogs.

**Domain and Forest Modes**

When you first install the Active Directory, it is possible to have NT 4 BDCs participate in the domain but obviously the Active Directory can offer far more than what was possible previously. To keep compatibility with the older NT 4 BDCs some of this functionality has to be disabled. This is known as running in mixed mode.

Once all domain controllers are Windows 2000 or above, you want to enable this new functionality by switching to an Active Directory native mode and with Windows 2000 these were the two domain options: mixed or native mode.

Windows 2003 Active Directory offers yet more abilities. So now, when all domain controllers in a domain are running Windows 2003 there is a Windows Server 2003 mode that enables the new abilities and once every domain in the entire forest is running Windows 2003 you can switch to a Windows 2003 forest mode.

Windows 2003 also introduced another domain and forest mode, known as "Windows 2003 Interim mode". This mode is available when upgrading from NT 4 to Windows 2003 directly and as we will see, overcomes some of the limitations of the Windows 2000 Active Directory implementation.

A brief overview of the changes between the various domain and forest modes is given overleaf. It is important to remember that switching to a higher domain mode is a one-way operation, you can never downgrade your mode: for example, you cannot go from native to mixed mode.
The Importance of a Domain

Domain Modes
Let's consider the domain modes:

Mixed Mode
This is the default domain mode when performing a fresh installation of Active Directory or when performing an upgrade and allows Windows 2000 and Windows 2003 domain controllers as well as NT 4 BDCs.

Windows 2000 Native Mode
In native mode, no NT 4 BDCs but only Windows 2000 and Windows 2003 domain controllers can be present.

This mode has additional functionality including nesting groups, Universal groups, and support for SID history and group conversions.

Windows Server 2003 Interim Mode
In 2003 interim mode, only Windows 2003 and NT 4 BDCs domain controllers (no Windows 2000 domain controllers) can be present in the domain.

This mode does not add any real extra functionality; it is used for the 2003 interim forest mode that fixed some problems with groups over certain sizes and site connectivity.

This domain mode can only be set when upgrading from Windows NT 4 to Windows 2003 and is set while running the DCPROMO utility on the first domain controller to be upgraded.

Windows Server 2003 Mode
In 2003 mode, only Windows 2003 domain controllers can be present. This has additional functionality over 2000 native mode, such as:

- Domain controller rename
- Password on InetOrgPerson objects
- Ability to redirect the default Users and Computers container
- Last logon timestamp attribute

Forest Modes
The forest modes:

Windows 2000
In Windows 2000 forest mode, all versions of domains and therefore all types of domain controllers (NT 4 BDC, 2000/2003 DC) are allowed. This is the default forest mode.

Windows Server 2003 Interim Mode
In Windows2003 interim mode, only Windows 2003 and NT 4 domain controllers can be present. This has additional functionality over the Windows 2000 mode including:
• More than 5000 users in a group via linked value replication (LVR)
The Importance of a Domain

- Improved ISTG (Inter Site Topology Generator), which is responsible for creating the replication topology between different locations
- Additional attributes added to Global Catalog

Windows Server 2003 Mode

In Server 2003 Mode, only Windows 2003 domain controllers can be present. This mode has additional functionality over 2003 Interim including:

- Dynamic auxiliary classes, which allow the creation of objects with an associated Time To Live (TTL) that are automatically removed once that time has expired
- The ability to convert User objects to INetOrgPerson (and vice versa)
- Schema de-/reactivation
- Domain rename
- Forest trusts
- Basic and Query-based groups
- 15 second intra-site replication frequency (with a 3 second offset)
- Linked Value Replication.

Linked Value Replication now allows individual elements of a multivalued attribute to be replicated instead of the whole value. This is a great improvement for Universal Group replication, which earlier required the replication of the entire group content each time a single change to its' membership occurred. This should also reduce the chances of normal group membership changes within a group. This can happen if administrators on different domain controllers modify the same group; one will overwrite the other. Now, the changes will be merged.

The overall goal of your environment is to get to Windows Server 2003 domain and forest modes, which opens up all the functionality in the most efficient way.

One vital point to understand is that forest mode restricts what versions of Domain Controllers can participate in the domain. They cannot be normal member servers or workstations. Even in a Windows Server 2003 mode domain, you can have NT 4 member servers and clients; the modes are only used to restrict the participating DCs so that all DCs can support the newly enabled functionality to stop any corruption of the database or variation in the service that clients receive.

Group Policy

Under Windows NT 4 domains, we had System Policies that could be modified and created with the supplied Group Policy Editor tool. A number of restrictions could be configured, which could be saved in a file called NTconfig.pol, placed in the netlogon
share on each domain controller, and applied when users logged on to the policy.
This was not very advanced, however, and all these policies were basically registry settings which 'tattooed' the registry—even after the policy was turned off, the registry change would still be in effect on the client machines.

A high level of granularity was also not possible. Policies could be applied to users, groups of users, or computers.

With the Active Directory, we no longer have System Policies, we have Group Policy, which really is a giant step past what we had under Windows NT 4, and its target is to meet the following ideal:

"The ability for the Administrator to state a wish about the state of their Users environment once, and then rely on the system to enforce that wish."

To achieve this, the Group Policy implementation (like the Active Directory) was rewritten from the ground up. While it still has a large portion based around the registry (although it no longer 'tattoos' the registry; if a policy setting is disabled or the policy deleted, all the changes it made are undone by default), group policies can also do much more including:

- Deploy applications on a per-user or machine basis in certain sites, domains or OUs; these deployed applications can be self-healing when using the Microsoft Installer Format (MSI files)
- Run logon/logoff/machine-startup/machine-shutdown scripts
- Redirect folders
- Configure local machine policies and rights
- Configure certificate, IP sec policies, etc
- Set the membership of local groups: for example, setting the members of the local Administrators group on machines to which the group policy is applied
- Enforce software restriction, which can prohibit certain applications from running (Quake can finally be wiped off your network!)

A Group Policy Object or GPO is a specific group of settings configured from the entire group policy scope: for example, it could contain some registry settings, some application deployment, and some folder redirection.
Group Policy Objects are split into two main branches, the User Configuration and the Computer Configuration, which, as the names suggest, apply to either the computer or the user.

Once the Group Policy Object is defined, it is then linked to a target container (since we are linking a GPO that can be linked to multiple containers while maintaining only one copy of the GPO itself).

GPOs can be linked to:

- A site
- A domain
- An organizational unit (at any level of the OU hierarchy)

The policies are applied in this order: first, the GPOs linked at the site the computer belongs to are applied, then the GPOs linked to the domain of the computer/user are applied, and then the GPOs linked to the OUs the computer/user are in are applied. For the Organizational Units, the GPOs at the top of the hierarchy are applied first and then the next layer until finally any GPOs linked at the actual OU the object is in are applied.

The GPO application is accumulative; you don't just get the policy 'closest' to the user/computer. It applies all of the policies but in case of a conflict the setting applied last takes precedence. There are some options that override this default behavior, but we will worry about that later.

You do not have to link policies at each level (in fact it is not that common to link GPOs at site levels) and you should try to minimize the various layers that GPOs are linked to, as this can adversely affect startup/logon times, as each GPO has to be processed.

Because the GPOs are linked, it is possible to actually link multiple GPOs at each applicable layer. The order of the application is set to ensure that the correct policy at each level takes precedence.

There is one other layer of policy; each machine has a local policy that can be modified using the Local Security Policy MMC snap-in. These settings are applied first, which means any setting applied via the Group Policy will override a local setting, which can be made on one of two local policies (Local Security Policy and the Local Group Policies).

The way to remember the application is LSDOU:

Local > Site > Domain > Organizational Unit
The Importance of a Domain

The application of group policy is shown in the following figure, which demonstrates policies applied at all the possible levels and the order in which they are applied:

In this figure, firstly, the local machine policy (1) is applied, then the group policy (2) linked at the site level. GPOs linked at the domain (3) are applied, followed by GPOs applied at the top layer of the OUs (4) and finally the GPOs are linked at the actual OU the object resides in (5).

Unlike NT 4 policies, which were applied at logon time, the Group Policy is actually refreshed periodically on the machines (every 90 minutes by default). However, the software deployment and folder redirection are not updated; if these were changed, it could have a very negative effect on your session.

Group Policy is extremely powerful and a great feature of the Active Directory, which you will undoubtedly use in your environment.

Summary

Under NT 4 domains there were not many design choices to make; the requirements dictated the technical solution. The Active Directory has many different architectural components with each having its own advantages and appropriate usage situations.

If your infrastructure was a house, the Active Directory would be its foundation. If your AD implementation is well designed, you will have a very strong foundation for everything that sits atop, implicitly driving everything towards a best-practice implementation. If your AD implementation is weak, then no matter what you do with the rest of your components, their foundation will be weak and in the long term require a lot more maintenance and sticky tape!
We saw in the schema section how many products modify the schema to store additional information pertinent to their products. However nearly all products (even those that do not modify the schema) utilize the Active Directory service in some fashion because it is the core building block of everything in your infrastructure.

More and more companies are leaning towards a Service-Orientated Architecture (SOA), which can benefit greatly from the Active Directory. During your analysis, therefore, it is vital to get everyone involved from the onset of the project and ensure that they know their participation is vital to the success of the project.

In the following chapters, we will emphasize on the importance of a thorough analysis of your current technical implementation and business structure. This is vital to ensure that the final design meets all the requirements in the most efficient way possible while having the flexibility to meet the needs of tomorrow without having to go back to the drawing board.

We have covered many principles in this chapter and have laid out the basics that will be explored in more detail, in our designs that follow. If there are concepts that are not crystal clear right now, don't panic, we'll be going over them again.
For more information: www.PacktPub.com/book/active_directory