Directory Service, LDAP and X.500

What is a directory service?

A directory is a specialized database optimized for reading, browsing and searching. Directories tend to contain descriptive, attribute-based information and support sophisticated filtering capabilities. Directories generally do not support complicated transaction or roll-back schemes found in database management systems designed for handling high-volume complex updates. Directory updates are typically simple all-or-nothing changes, if they are allowed at all. Directories are tuned to give quick response to high-volume lookup or search operations. They may have the ability to replicate information widely in order to increase availability and reliability, while reducing response time. When directory information is replicated, temporary inconsistencies between the replicas may be okay, as long as they get in sync eventually.

There are many different ways to provide a directory service. Different methods allow different kinds of information to be stored in the directory, place different requirements on how that information can be referenced, queried and updated, how it is protected from unauthorized access, etc. Some directory services are *local*, providing service to a restricted context (e.g., the finger service on a single machine). Other services are global, providing service to a much broader context (e.g., the entire Internet). Global services are usually *distributed*, meaning that the data they contain is spread across many machines, all of which cooperate to provide the directory service. Typically a global service defines a uniform *namespace* which gives the same view of the data no matter where you are in relation to the data itself. The Internet Domain Name System (DNS) is an example of a globally distributed directory service.

1.2. What is LDAP?

LDAP stands for Lightweight Directory Access Protocol. As the name suggests, it is a lightweight protocol for accessing directory services, specifically X.500-based directory services. LDAP runs over TCP/IP or other connection oriented transfer services. The nitty-gritty details of LDAP are defined in RFC2251 "The Lightweight Directory Access Protocol (v3)" and other documents comprising the technical specification RFC3377. This section gives an overview of LDAP from a user's perspective.

What kind of information can be stored in the directory? The LDAP information model is based on *entries*. An entry is a collection of attributes that has a globally-unique Distinguished Name (DN). The DN is used to refer to the entry unambiguously. Each of the entry's attributes has a *type* and one or more *values*. The types are typically mnemonic strings, like "cn" for common name, or "mail" for email address. The syntax of values depend on the attribute type. For example, a cn attribute might contain the value Babs Jensen. A mail attribute might contain the value "babs@example.com". A jpegPhoto attribute would contain a photograph in the JPEG (binary) format.

How is the information arranged? In LDAP, directory entries are arranged in a hierarchical tree-like structure. Traditionally, this structure reflected the geographic and/or organizational boundaries. Entries representing countries appear at the top of the tree. Below them are entries representing states and national organizations. Below them might be entries representing organizational units, people, printers, documents, or just about anything else you can think of. Figure 1.1 shows an example LDAP directory tree using traditional naming.

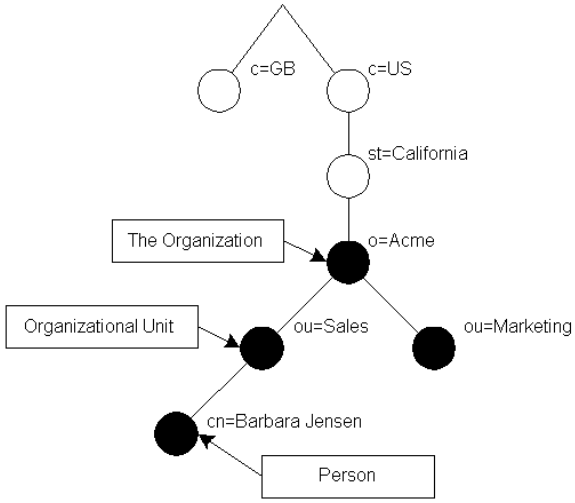


Figure 1.1: LDAP directory tree (traditional naming)

The tree may also be arranged based upon Internet domain names. This naming approach is becoming

increasing popular as it allows for directory services to be located using the *DNS*. Figure 1.2 shows an example LDAP directory tree using domain-based naming.

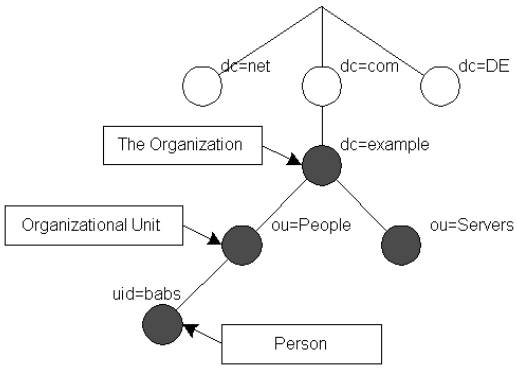


Figure 1.2: LDAP directory tree (Internet naming)

In addition, LDAP allows you to control which attributes are required and allowed in an entry through the use of a special attribute called objectClass. The values of the objectClass attribute determine the *schema* rules the entry must obey.

How is the information referenced? An entry is referenced by its distinguished name, which is constructed by taking the name of the entry itself (called the Relative Distinguished Name or RDN) and concatenating the names of its ancestor entries. For example, the entry for Barbara Jensen in the Internet naming example above has an RDN of uid=babs and a DN of uid=babs,ou=People,dc=example,dc=com. The full DN format is described in RFC2253, "Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names."

How is the information accessed? LDAP defines operations for interrogating and updating the directory. Operations are provided for adding and deleting an entry from the directory, changing an existing entry, and changing the name of an entry. Most of the time, though, LDAP is used to search for information in the directory. The LDAP search operation allows some portion of the directory to be searched for entries that match some criteria specified by a search filter. Information can be requested from each entry that matches the criteria.

For example, you might want to search the entire directory subtree at and below dc=example,dc=com for people with the name Barbara Jensen, retrieving the email address of each entry found. LDAP lets you do this easily. Or you might want to search the entries directly below the st=California,c=US entry for organizations with the string Acme in their name, and that have a fax number. LDAP lets you do this too. The next section describes in more detail what you can do with LDAP and how it might be useful to you.

How is the information protected from unauthorized access? Some directory services provide no protection, allowing anyone to see the information. LDAP provides a mechanism for a client to authenticate, or prove its identity to a directory server, paving the way for rich access control to protect the information the server contains. LDAP also supports privacy and integrity security services.

1.3. How does LDAP work?

LDAP directory service is based on a *client-server* model. One or more LDAP servers contain the data making up the directory information tree (DIT). The client connects to servers and asks it a question. The server responds with an answer and/or with a pointer to where the client can get additional information (typically, another LDAP server). No matter which LDAP server a client connects to, it sees the same view of the directory; a name presented to one LDAP server references the same entry it would at another LDAP server. This is an important feature of a global directory service, like LDAP.

1.4. What about X.500 (DAP)?

Technically, LDAP is a directory access protocol to an X.500 directory service, the OSI directory service. Initially, LDAP clients accessed gateways to the X.500 directory service. This gateway ran LDAP between the client and gateway and X.500's Directory Access Protocol (DAP) between the gateway and the X.500 server. DAP is a heavyweight protocol that operates over a full OSI protocol stack and requires a significant amount of computing resources. LDAP is designed to operate over TCP/IP and provides most of the functionality of DAP at a much lower cost.

While LDAP is still used to access X.500 directory service via gateways, LDAP is now more commonly directly implemented in X.500 servers.

The stand-alone LDAP daemon, or *slapd*(8), can be viewed as a *lightweight* X.500 directory server. That is, it does not implement the X.500's DAP. As a *lightweight directory* server, *slapd*(8) implements only a subset of the X.500 models.

If you are already running a X.500 DAP service and you want to continue to do so, you can probably stop reading this guide. This guide is all about running LDAP via *slapd*(8), without running X.500 DAP. If you are not running X.500 DAP, want to stop running X.500 DAP, or have no immediate plans to run X.500 DAP, read on.

It is possible to replicate data from an LDAP directory server to a X.500 DAP DSA. This requires an LDAP/DAP gateway. OpenLDAP does not provide such a gateway, but our replication daemon can be used to replicate to such a gateway. See the Replication with slurpd chapter of this document for information regarding replication.

Source:

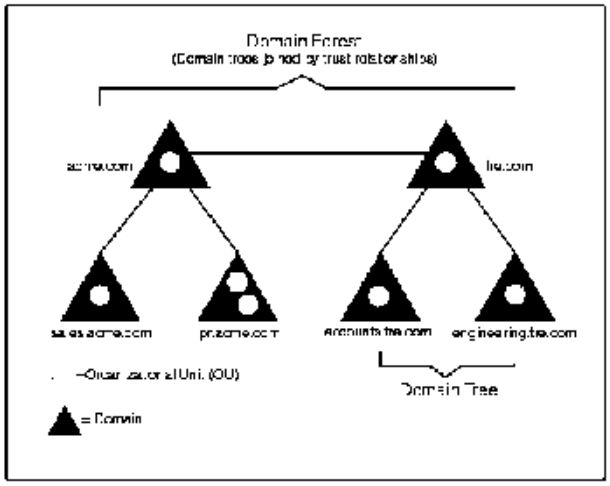
http://www.openldap.org/doc/admin22/intro.html

Tree and Forest

The Domain is the core unit of logical structure in Active Directory. All objects that share a common directory database and trust relationship with other domain and security policies are known as Domains. Each domain stores information only about the objects that belong to that domain.

All security polices and settings, such as administrative rights, security policies, and Access Control Lists (ACLs), do not cross from one domain to another. Thus, a domain administrator has full rights to set policies only within domain they belong to.

Domains provide administrative boundaries for objects and manage security for shared resources and a replication unit for objects.



A Tree

Trees are collections of one or more domains that allow global resource sharing. A tree may consist of a single domain or multiple domains in a contiguous namespace. A domain added to a tree becomes a child of the tree root domain. The domain to which a child domain is attached is called a parent domain. A child domain can also have its multiple child domains. Child domain uses the name then its parent domain name and gets a unique Domain Name System (DNS).

For example, if tech.com is the root domain, users can create one or more Child domains to tech.com such as north.tech.com and or south.tech.com. These “children” may also have child domains created under them, such as sales.north.tech.com.

The domains in a tree have two way, Kerberos transitive trust relationships. A Kerberos transitive trust simply means that if Domain A trusts Domain B and Domain B trusts Domain C, then Domain A trusts Domain C. Therefore, a domain joining a tree immediately has trust relationships established with every domain in the tree.

A Forest

A forest is a collection of multiple trees that share a common global catalog, directory schema, logical structure, and directory configuration. Forest has automatic two way transitive trust relationships. The very first domain created in the forest is called the forest root domain.

Forests allow organizations to group their divisions that use different naming schemes and may need to operate independently. But as an organization, they want to communicate with the entire organization via transitive trusts and share the same schema and configuration container.

Source:

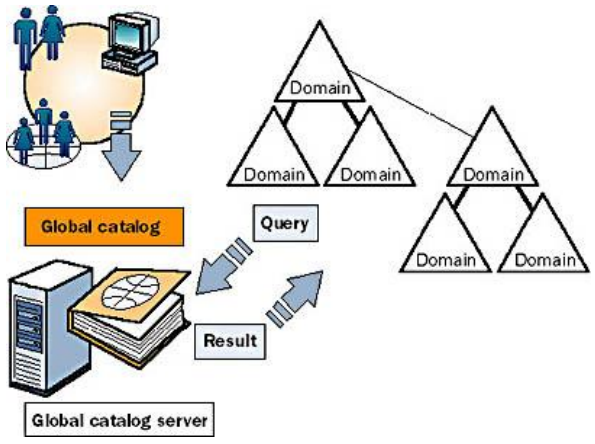
http://www.tech-faq.com/tree-and-forest-in-active-directory.html

Global Catalog

Domains and Forests can also share resources available in active directory. These resources are searched by Global Catalog across domains and forests and this search is transparent to user. For example, if you make a search for all of the printers in a forest, this search goes to global catalog server for its query and then global catalog returns the results. Without a global catalog server this query needs to go to every domain in the forest of its result.

It is important to have a global catalog on at least one domain controller because many applications use port 3268 for searching. For example, if you do not have any global catalog servers in your network, the Search command on the Start menu of Windows 2000/2003 cannot locate objects in Active Directory.

The global catalog is a domain controller that contains attributes for every object in the Active Directory. By default, only the members of the Schema Admins group have rights to change which attributes stored in the global catalog, according to organization's requirements.



The global catalog contains:

- The commonly used attributes need in queries, such as a user's first and last name, and logon name.
- All the information or records which are important to determine the location of any object in the directory.
- A default subset of attributes for each object type.
- All the access related permissions for every object and attribute that is stored in the global catalog. Say, without permission you can't access or view the objects. If you are searching for an object where you do not have the appropriate permissions to view, the object will not appear in the search results. These access permissions ensure that users can find only objects to which they have been assigned access.

A global catalog server is a domain controller that contains full and writable replica of its domain directory, and a partial, read-only replica of all other domain directory partitions in the forest. Let's take an example of a user object; by default user objects have lot of attributes such as first name, last name, address, phone number, and many more. The Global Catalog will store only the main attributes of user objects in search operations like a user's first name and last name, or login name. This partial attributes of that user object which is stored would be enough to allow a search for that object to be able to locate the full replica of the object in active directory. If a search comes to locate objects, then first it goes to local global catalog and reduces network traffic over the WAN.

Domain Controllers always contain the full attribute list for objects belonging to their domain. If the Domain Controller is also a GC, it will also contain a partial replica of objects from all other domains in the forest.

It is always recommended to have a global catalog server for every active directory site in an enterprise network.

Source:

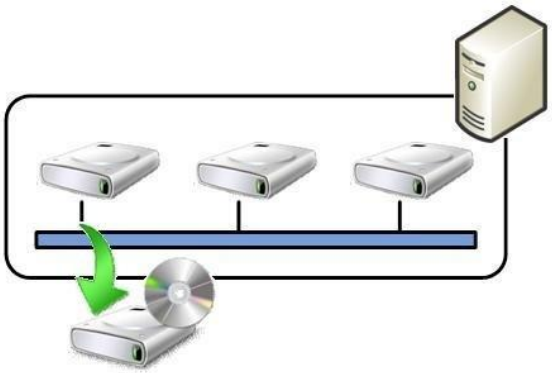
http://www.tech-faq.com/global-catalog-in-active-directory.html

Directory Backup and Restore

An Overview on Backing up and Restoring Active Directory

To ensure availability of mission critical resources and network objects, and business continuity, you would need to perform back ups of Active Directory if it is running in your environment. This is because Active Directory normally hosts mission critical data, and resources. Backups are typically preformed for a number of reasons, including the following:

- **Protect your network environment from the accidental deletion of, or modification of data, and from hardware failures**: Having a readily accessible back up of Active Directory would ensure that you can recover any important Active Directory objects which were deleted in error. Backups also prove invaluable when unauthorized users intentionally delete or modify data. The backup would enable you to restore data to its previous state of integrity. Because certain hardware failures such as corrupted hard disk drives can cause considerable loss of data, backing up your data would ensure that the business can continue to perform its mission critical functions when such an event does occur.
- **Store mission critical data**: It is recommended to regularly back up mission critical data so that any previous version of information can be accessed, if necessary, at some time in the future.



Because Active Directory is dependant on the Registry, you need to back up files within the system directory. These files are called system files. **System state data** basically contains the main configuration information in Windows 2000, and Windows Server 2003. What actual information is included in system state data is determined by operating system (OS) configuration. System state typically includes the following important data, files and components:

- The Windows Registry
- The contents of the SYSVOL directory
- Files which are protected by the Windows File Protection system
- Boot and system files: Ntdetect.com, Ntldr and Bootsect.dat.
- The COM+ Class Registration database
- The Active Directory database (Ntds.dit), including all log files and checkpoint files
- Cluster service files
- Certificate service files
- The Internet Information Server (IIS) metabase

You can use one of the methods listed below to back up Active Directory.

- You can back up the system state data only
- You can back up Active Directory as part of a full system backup
- You can back up Active Directory as part of a partial system backup

The best option to use when specifying what data or components should be backed up in the Active Directory backup; is to specify a back up of system state data. This ensures that all core system files are backed up. When a full system backup is performed, system state data is automatically included in the back up process. When performing a partial backup, you can specify that system state data should be included. Manually specifying individual files and components for an Active Directory backup can be an extremely complicated process. Apart from having to be able to identify and specify all important system files and components, you also need to be able to specify which other important Active Directory data and components need to be

backed up, such as the replication topology, and Group Policy information.

You can back up Active Directory by using the Windows Server 2003 Backup utility, or you from the command line, using the Ntdsutil command-line utility. The Windows Server 2003 Backup utility includes the feature of using **volume shadow copying** to back up open files. With the previous versions of Windows, a third party backup tool had to be used to back up open files. The Volume Shadow Copy service creates a read-only copy of any open files. This in turn ensures that these files can continue to be accessed.

In Windows 2000 Active Directory, you could only perform one of the following restore methods:

- Authoritative Restore
- Non- Authoritative

When it comes to **restoring Windows Server 2003 Active Directory**, you can use one of the following restore methods:

- **Normal Restore**: In Windows 2000, this was your Non-Authoritative restore method. A Normal restore functions pretty much the same as a Non-Authoritative restore. With a Normal restore, the Backup utility is run on the computer while in Directory Services Restore Mode. After the domain controller is rebooted, normal replication occurs with replication partners.

A normal restore is typically performed when the following conditions exist:

-
  - A domain has multiple domain controllers, and only one domain controller is operational. You can use a Normal restore to restore all other domain controllers in the domain.
  - A domain has a single domain controller, and that domain controller has to be restored. You can also choose to alternatively perform a Primary restore of Active Directory.

- **Authoritative Restore**: An Authoritative restore of Active Directory has to be performed in cases where a Normal restore would not be able to return Active Directory to the correct state. For instance, if an organizational unit was deleted in error, a Normal restore would only result in the particular OU being deleted once again, after replication. This is basically due to the replication partners having a higher version number for the particular OU. An Authoritative restore has a similar process to that of a Normal restore, the difference being that after system data is restored, you define certain Active Directory objects as being authoritative. When Active Directory objects are defined as authoritative, the particular objects have the higher version numbers. This results in these objects being replicated to the other domain controller's copies of the Active Directory database.
- **Primary Restore**: The Primary restore method is used when each domain controller within a domain hosting multiple domain controllers, needs to be restored. What this means is that the entire domain has to be reconstructed from the Active Directory backup. This method can also be used to restore Active Directory for a domain that only has one domain controller. The Primary restore method is selected in Windows Server 2003 Backup utility by merely enabling the Primary restore method checkbox. This removes previous complexities associated with performing this type of restore in Windows 2000. The Primary restore process is also very similar to that performed for a Normal restore of Active Directory.

Understanding Trust Relationship

In the Windows NT domain model, domains had to be bound together through trust relationships simply because the SAM databases used in those domains could not be joined. What this meant was that where a domain trusted another Windows NT domain, the members of the domain could access network resources located in the other domain. Defining trust relationships between domains

eliminates the need for an Administrator to configure user accounts in multiple domains.

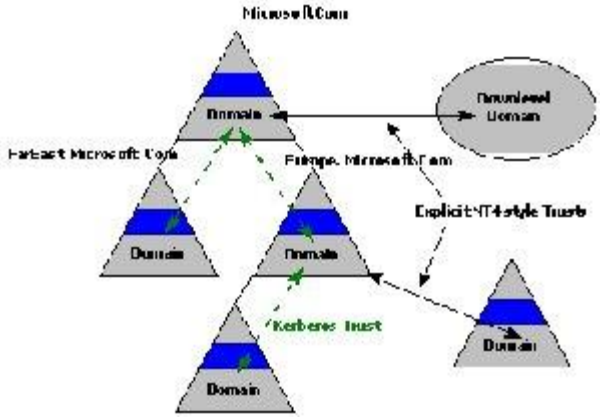
In a trust relationship, the two domains are referred to as the trusting domain and the trusted domain. The trusted domain is the domain where the trust relationship is created. The trusting domain is the other domain specified in the trust, that is, the one wherein network resources can to be accessed. The trusting domain in this case recognizes the logon authentications of the trusted domain. The NT LanMan Challenge Response supports the logon trust relationship. This allows pass-through authentication of users from the trusted domain. One of the shortfalls of Windows NT trust relationships is that trusts between domains were one way and non-transitive. This meant that the defined trust relationship ended with the two domains between which the particular trust was created. The rights implicit in the trust relationship also flowed only in one direction. Because of this, defining and managing trust relationships in the Windows NT domain structure was a cumbersome and labor intensive task. The Windows NT domain worked well in small enterprises where one domain typically existed in the enterprise. In those larger enterprises that have multiple domains, Administrators have to define trust relationships between the domains in order for a user in one domain to access resources in another domain.

In Windows 2000 and Windows 2003, Active Directory is built on the concept of trust relationships between domains. Although the actual concept of trust relationships is not new in Windows Server 2003, there are new trust capabilities and trust types available for Windows Server 2003 Active Directory domains.

In Windows Server 2003, authentication of users or applications occurs through the use of one of the following trust protocols:

- NT LAN Manager (NTLM) protocol: This protocol is used when one of the computers in the trust relationship does not support the Kerberos version 5 protocol.
- The Kerberos version 5 protocol is the default trust protocol used when computers in trust relationships are running Windows Server 2003.

Domain Trust Relationships



The characteristics of Windows Server 2003 trusts are outlined below:

- Trusts can be non-transitive or transitive:
  - Transitive trusts: With transitive trusts, trust is applicable for each trusted domain. What this means is where Domain1 trusts Domain2, and Domain2 trusts Domain3, Domain1 would also trust Domain3.
  - Non-transitive trust: The defined trust relationship ends with the two domains between which the particular trust is created.
- Trusts can be one way or two way:
  - One way trusts: Based on the direction of the trust, one way trust can further be broken into either incoming trust or outgoing trusts. One way trust can be transitive or non-transitive:
    - Incoming Trust: With incoming trust, the trust is created in the trusted domain and users in the trusted domain are able to access network resources in the trusting domain or other domain. Users in the other domain cannot however access network resources in the trusted domain.
    - Outgoing Trust: In this case, users in the other domain can access network resources in the initiating domain. Users in the initiating domain are not able to access any resources in the other domain.

    - Two way trusts: A two way trust relationship means that where Domain1 trusts Domain2, then Domain2 trusts Domain1. The trust basically works both ways and users in each domain are able to access network resources in either one of the domains. A two way, transitive trust relationship is the trust that exists between parent domains and child domains in a domain tree. In two way transitive trust, where Domain1 trusts Domain2 and Domain2 trusts Domain3, then Domain1 would trust Domain3 and Domain3 would trust Domain1.Two way transitive trust is the default trust relationship between domains in a tree. It is automatically created and exists between top level domains in a forest.
- Trusts can be implicit or explicit trusts:
    - Implicit: Automatically created trust relationships are called implicit trust. An example of implicit trust is the two way transitive trust relationship that Active Directory creates between a parent and child domains.
    - Explicit: Manually created trust relationships are referred to as explicit trust.

Types of Active Directory Trust Relationships

The types of trust relationships that can be created and configured for Active Directory domains are discussed in this section. As an Administrator for Active Directory Windows Server 2003 domains, it is important to understand the different types of trusts that are supported in Windows Server 2003 and to know which trust relationship to create for the different network resource access requirements that exist within the organization.

- **Tree-root trust**: Tree root trust is automatically/implicitly created when a new tree root domain is added to a forest. The trust relationship exists between two root domains within the same forest. For instance, if there is an existing forest root domain, and a new tree root domain is added to the same forest, tree root trust is formed between the new tree root domain and the existing forest root domain. Tree root trust is transitive and two way.
- **Parent-child trust**: Parent-child trust is implicitly established when new child domains are added to a domain tree. Parent-child trust is a two-way, transitive trust relationship. Active Directory automatically creates a trust relationship between the new child domain and the domain directly above it in the domain namespace hierarchy. What this means is that the trust relationship exists between those domains that have a common contiguous DNS namespace and who are part of the same forest. Parent-child trust enables child domain authentication requests to be passed through the parent domain for authentication. In addition, when a new domain is added to the tree, trust relationships are created with each domain in the tree. This means that network resources in the tree's individual domains can be accessed by all other domains in the tree.
- **Shortcut trust**: An administrator explicitly creates a shortcut trust and is either a one way transitive trust or two way transitive trust. Shortcut trust is usually created when users want to speed up or enhance authentication performance between two domains in different trees but within the same forest. One way shortcut trust should be created when users in Domain1 need to access Active Directory objects in Domain2 but users in Domain2 do not need to access objects in Domain1. Two way shortcut trust should be created when users in each domain need to access objects in each other's domain.
- **Realm trust**: An administrator explicitly creates realm trust and it can be defined as either a transitive or non-transitive trust. It can also either be a one way or two way trust. Realm trust enables users to create a trust relationship between a Windows Server 2003 Active Directory domain and a non-Windows Kerberos version 5 realm. Realm trust therefore facilitates interoperability between a Windows Server

2003 domain and a realm used in Kerberos version 5 implementations.

- **External trust**: An administrator explicitly defines the external trust to enable trust between domains that are located in different forests and to create trust between an Active Directory domain and a down-level Windows NT 4 domain. External trust is always non-transitive but can be either one-way trusts or two-way trusts. External trust is usually only created in Windows Server 2003 Active Directory environments when users need to access network resources in a domain that resides in a different forest and forest trust cannot be created between the two domains. When external trust is created between an Active Directory domain and a down-level Windows NT 4 domain, it is a one-way, non-transitive trust relationship.
- **Forest trust**: An Administrator explicitly created Forest trust to enable trust between two Active Directory forests. Forest trust is transitive in nature and can either be one-way or two-way. Forest trust is only available in Windows Server 2003. Before users can create forest trust between two forests, each domain in the particular forest and each forest has to be raised to and run at the Windows Server 2003 functional level. Because forest trust is created between two root domains of two forests, it can create two way trusts with each domain within the two forests. This basically means that users would be able to access Active Directory objects between all domains encompassed by the particular forest trust relationship.

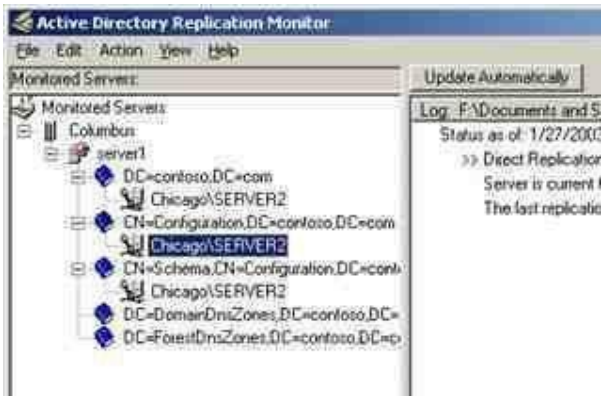
Source:

http://www.tech-faq.com/understanding-trust-relationships.html

Directory Replication

The initial Windows NT versions were designed as single master network environments. The primary domain controller (PDC) was responsible for managing the domain database's master copy. The PDC was therefore responsible for replicating any changes to the backup domain controllers (BDCs). In these environments, any changes had to be performed on the PDC, which then replicated these database changes to the BDCs. What this meant was that in cases where the PDC was unavailable, no changes were made to the domain database. From this simple discussion, it is clear that the single master environment of the earlier Windows NT versions had a limitation when it came to reliability and continuously ensuring that changes could be made to the domain database.

In most network environments, more than one domain controller has to exist to provide fault tolerance and improve reliability and performance. Fault tolerance is present when business continuity exists when one domain controller fails because the other domain controller(s) in the environment supplies network resources. Having multiple domain controllers in a network environment improves performance because the processing load can be distributed to all domain controllers.



Active Directory differs from the design of the earlier Windows NT domain environments because it is a scalable, distributed multi-master replicated database. Information on network resources within the organization is stored in the Active Directory database. In addition to this, all domain controllers host a full replica of the domain information for its own domain. Domain controllers in Windows 2000 and Windows Server 2003 environments hold a read/write copy of the Active Directory database. Domain controllers in these environments therefore maintain and manage the replica of all Active Directory objects (network resources) located in the domain to which it is a member of.

In Windows 2000 and Windows Server 2003 environments, in Active Directory terminology, each

domain controller contains a full copy of its own directory partition. Another term used to refer to directory partition is naming context. In Active Directory environments, a directory tree contains all Active Directory objects in the forest. A forest is the grouping of two or more domain trees or domains that do not have a common contiguous namespace. That is, they have non-contiguous namespaces. In Active Directory, the directory tree is partitioned. This enables portions of the tree to be distributed to domain controllers in other domains in the forest. The copy of the directory partition that holds all the attributes for each directory partition object is called a replica. The replica on each domain controller has read and write attributes.

In Active Directory, changes can be made to the Active Directory database on any domain controller within the Active Directory environment. To overcome the limitations of the Windows NT domain environments illustrated earlier, each domain controller must include all information that is created or changed on any other domain controller. Active Directory replication ensures that the information or data between domain controllers remains updated and consistent. Replication is the process that ensures that changes made to a replica on one domain controller are transferred to replicas on the remainder of the domain controllers. It is Active Directory replication that ensures that Active Directory information that domain controllers host is synchronized.

Active Directory's multi-master environment eliminates the domain controllers as single points of failure because an Administrator can perform changes to the Active Directory database on any domain controller and these changes are replicated to the other domain controllers within the domain.

What Information is Replicated in Active Directory

In Active Directory, there are certain actions that are considered *Active Directory replication triggers.* The activities that trigger or initiate Active Directory replication are summarized below:

- When an object is *created.*
- When an object is *deleted.*
- When an object is *moved.*
- When an object is *changed* or modified.

Domain controllers typically contain the following directory partition replicas or naming context replicas:

- *Configuration:* The configuration partition or naming context (NC) contains objects that relate to the logical structure of the forest, structure of the domain, and replication topology. Each domain controller in the forest contains a read/write copy of the configuration partition. Any *objects stored in the configuration partition are replicated to each domain controller in each domain and in a forest.*
- *Domain:* The domain partition or naming context (NC) contains all objects that are stored in a domain. Each domain controller in a domain has a read/write copy of the domain partition. *Objects in the domain partition are replicated to only the domain controllers within a domain.*
- *Schema:* The schema partition or naming context (NC) contains objects that can be created in the Active Directory and the attributes that these objects can contain. Domain controllers in a forest have a read-only copy of the schema partition. *Objects stored in the schema partition are replicated to each domain controller in domains/forests.*
- *Application:* The application partition is a new feature introduced in Windows Server 2003. This partition contains application specific objects. The objects or data that applications and services store here can comprise of any object type excluding security principles. Security principles are Users, Groups, and Computers. The application partition typically contains DNS zone objects and dynamic data from other network services such as Remote Access Service (RAS) and Dynamic Host Configuration Protocol (DHCP).

Soruce:

http://www.tech-faq.com/active-directory-replication.html

Schema Master

The schema master controls all originating updates to the schema. The schema contains the master list of object classes and attributes that are used to create all AD DS objects, such as computers, users, and printers. The domain controller that holds the schema master role is the only domain controller that can perform write operations to the directory schema. These schema updates are replicated from the schema operations master to all other domain controllers in the forest.

Operational Models

Single Domain Model

The most basic of all Active Directory structures is the single domain model; this type of domain structure comes with one major advantage over the other models: simplicity. A single security boundary defines the borders of the domain, and all objects are located within that boundary. The establishment of trust relationships between other domains is not necessary, and implementation of technologies such as Group Policies is made easier by the simple structure. Many organizations that have lived with a multiple domain NT structure may think that they cannot consolidate on a single domain model. However, more organizations than not can take advantage of this design because Active Directory has been simplified and its ability to span multiple physical boundaries has been enhanced.

Choosing the Single Domain Model

The single domain model is ideal for many organizations and can be modified to fit many more. A single domain structure possesses multiple advantages, first and foremost being simplicity. As any administrator or engineer who has done work in the trenches can contest, often the simplest design works the best. Adding unnecessary complexity to systems' architecture introduces potential risk and makes troubleshooting these systems more difficult. Consequently, consolidating complex domain structures such as NT 4.0 into a simpler single domain Active Directory structure can reduce the costs of administration and minimize headaches in the process.

Another advantage that is realized by the creation of a single domain is the attainment of centralized administration. Many organizations with a strong central IT structure want the capability to consolidate their control over their entire IT and user structure. NT domains were notoriously lacking in their capability to scale to these levels, and the types of central control that organizations wanted were not available. Active Directory and, specifically, the single domain model allows for a high level of administrative control and the ability to delegate tasks to lower sets of administrators. This has proven to be a strong draw to Active Directory.

Not all Active Directory structures can be composed of a single domain, however, and some factors may limit an organization's ability to adopt a single domain structure. If these factors affect your organization, you might need to begin expanding your domain model to include other domains in the forest and a different domain design. For example, the single security boundary formed by a single domain may not be exactly what your organization needs. Although OUs can be used to delegate administration of security elements, the domain itself is the security boundary in Active Directory structures. If the security lines within your organization need to follow exact boundaries, a single domain may not be for you. For example, if your HR department requires that no users from IT have access to resources within its environment, you will need to expand your domain structure to accommodate the additional security concerns.

Another disadvantage of the single domain model is that a single domain in a forest necessitates that the computer with the role of schema master is located in that domain. This places the schema master within the domain that contains all the user accounts. Although access to the schema master can be strictly controlled through proper administration, your risk of schema exposure is greater when the schema master role resides in a user domain. If this design model poses problems for you as an organization, design models that separate the schema master into a placeholder domain can do the trick. The placeholder domain model is described in more detail later in this chapter.

Real-World Design Example

To illustrate a good example of an organization that would logically choose a single domain model, let's consider fictional Company A. Company A is a 500-

user organization with a central office located in Minneapolis. A few smaller branch offices are scattered throughout the Midwest, but all help desk administration is centralized at the company headquarters. Company A currently utilizes a single NT user domain and has multiple resource domains in various locations across the country.

The IT team in Minneapolis is designing an Active Directory structure and wants to centralize administration at corporate headquarters. Branch offices should have the ability to change passwords and clear print jobs locally, but should have no other form of administrative privilege on the network.

During the Active Directory design process, Company A started with a single Active Directory forest, domain, and namespace named companya.net. Organizational units for each branch office were added so as to delegate password-change control and print administration to those offices.

Current NT 4.0 domains were consolidated into the Active Directory structure, as shown in Figure 5.3. Company A could not justify the existence of additional domains because their security model was centralized, and it did not have any far-flung geographical locations with slow link speeds to the main office or any other similar constraints that required additional domains.

Figure 5.3 Active Directory structure with organizational unit structure.

Delegation of password-change control and other local administrative functions was granted to individuals in each specific geographical OU, which gave those administrators permissions specific to only resources within their own group but maintained central administrative control in Minneapolis. A detailed discussion of organizational unit design is covered in Chapter 6, "Designing Organizational Unit and Group Structure."

Several Active Directory sites were created to control the frequency of replication. A site was positioned to correspond with each separate geographical area, creating a site structure similar to the one shown in Figure 5.4.

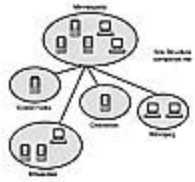


Figure 5.4 Site structure created by geographical locations.

Creating the separate sites helped to throttle replication traffic and reduce the load placed on the WAN links between the sites. For more details about site links and replication, see Chapter 7, "Active Directory Infrastructure."

This type of single domain design is ideal for the type of organization described in this section and actually can be used for many other types of organizations, large and small. Because delegation of administration is now accomplished through the use of OUs and Group Policy objects, and the throttling of replication is accomplished through AD sites, the number of reasons for organizations to use multiple domains has been reduced.

Multiple Subdomain Model

For various reasons, organizations may need to add more than one domain to their environment but preserve the functionality that is inherent in a single forest. When this occurs, the addition of one or multiple subdomains into the forest is warranted. Domain addition should not be taken lightly, however, and proper consideration must be given to the particular characteristics of multiple domain models.

By default, two-way transitive trusts exist between subdomain and domain in Active Directory. Bear in mind, however, that this does not mean that resource access is auto-matically granted to members of other domains. A user in subdomain B is not automatically granted any rights in domain A; they need to be explicitly defined through the use of groups. Understanding this concept will help to determine the logistics of domain addition.

When to Add Additional Domains

As previously mentioned, it is advisable to begin your Windows .NET Active Directory design with a single domain and then add domains only when absolutely necessary. Adding child domains to your existing domain structure may become necessary if the following traits exist within your infrastructure:

- **Decentralized Administration**—If different branches of your organization generally manage their own IT structure and there are no future plans to consolidate them into a centralized model, multiple interconnected domains may be ideal. Each domain acts as a security boundary for most types of activity and can be set up to disallow administration from escaping the boundaries of domains. This approach operates in much the same way as NT domains and inherits many of the limitations associated with them as well. In other words, it is better to try to centralize administration before deploying Active Directory because you will gain more of AD's advantages.
- **Geographic Limitations**—If extremely slow or unreliable links or great geographical distances separate different parts of your company, it may be wise to segment the user population into separate groups. This will help to limit replication activity between domains and also make it easier to provide worktime support for distant time zones. Keep in mind that slow links by themselves do not necessitate the creation of multiple domains, as Windows .NET Active Directory uses the concept of Active Directory sites to throttle replication across slow links. The main reason that might exist for domain creation for geographical reasons is administrative flexibility. In other words, if there is a problem with the network in Japan, a Japanese administrator will be able to have more power to administer the Asia domain and will not need to call the North American administrator in the middle of the night.
- **Unique DNS Namespace Considerations**—If two organizational entities want to use their Internet-registered namespace for Active Directory but use a common forest, such as hotmail.com or microsoft.com, those domains must be added as separate domains. This type of domain model is described more fully in the section "Multiple Trees in a Single Forest Model" later in this chapter.
- **Special Password Policies**—Because password policies are set on a domain level, if any specific password policies must be set differently between domains, separate domains must be set up to segregate those policies. This is rarely a real-life design issue that by itself creates a new domain, but knowledge of this limitation will help in the design process.
- **Enhanced Security Concerns**—Depending on the needs of your organization, separating the schema master role into a domain separate from your users may be applicable. In this case, the single domain model would not be applicable, and a model such as the peer-root or placeholder domain would be more appropriate.

When you're contemplating additional domains, remember the mantra "Simplicity is best." However, if during the design process, the specific need arises to add domains, proper design is still warranted, or your environment will run the risk of looking like the type of messed-up NT domain structure that you have been trying to avoid.

Real-World Design Example

To illustrate an organization that would have grounds to establish multiple domains, we'll use the following example. Company B is an engineering company based in York, Pennsyl-vania. Administration for all branch locations is currently centralized in the home office, and OUs and Group Policies are used for delegation of lower-level tasks. Recently, the company acquired two separate companies named Subsidiary A and Subsidiary B; each contains its own IT departments and operates in separate geographical areas. Company B decided to implement Active Directory as part of a Windows .NET Server 2003 implementation and wanted to include the two acquired companies into a single common forest.

Because of the fact that each acquired company possesses its own IT department and there are no immediate plans to consolidate those functions centrally, Company B decided to deploy an Active Directory structure with two subdomains for Subsidiary A and Subsidiary B, as shown in Figure 5.5.

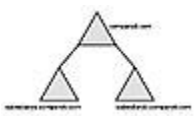


Figure 5.5 Active Directory with two subdomains.

This design model allowed for a certain degree of administrative freedom with the newly acquired subsidiaries but also allowed for a common forest and schema to be used and kept the domains within the same DNS namespace.

This design model has the particular advantage of being politically easier to implement than consolidation of existing domains. Branch offices and subsidiary companies can keep their own domain structure and security boundaries, and their IT teams can retain a greater deal of administrative autonomy. In other words, the NT domain administrator from a branch office won't be extremely upset after you've taken away his domain during or after a migration.

Be warned, however, that consolidation of NT domains into fewer domains is a key feature of Active Directory, so the addition of domains purely for political reasons adds complexity and potentially unnecessary infrastructure. It is therefore very important to consider the alternatives before deciding on this design model.

Multiple Trees in a Single Forest Model

Let's say that your organization would like to look at Active Directory and wants to use an external namespace for your design. However, your environment currently uses multiple DNS namespaces and needs to integrate them into the same design. Contrary to popular misconception, integration of these namespaces into a single AD forest can be done through the use of multiple trees that exist in one forest. One of the most misunderstood characteristics of Active Directory is the difference between a contiguous forest and a contiguous DNS namespace. Many people do not realize that multiple DNS namespaces can be integrated into a single Active Directory forest as separate trees in the forest. For example, Figure 5.6 shows how Microsoft could theoretically organize several Active Directory domains that share the same forest but reside in different DNS namespaces.

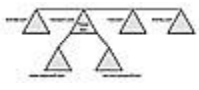


Figure 5.6 Sample Active Directory forest with multiple unique trees within the same forest.

Only one domain in this design is the forest root, in this case microsoft.com, and only this domain controls access to the forest schema. All other domains, including subdomains of microsoft.com and the other domains that occupy different DNS structures, are members of the same forest. All trust relationships between the domains are transitive, and trusts flow from one domain to another.

When to Choose a Multiple Tree Domain Model

If your organization currently operates multiple units under separate DNS namespaces, one option may be to consider a design such as this one. It is important to understand, however, that simply using multiple DNS namespaces does not automatically qualify you as a candidate for this domain design. For example, you could own five separate DNS namespaces and instead decide to create an Active Directory structure based on a new namespace that is contiguous throughout your organization. Consolidating your Active Directory under this single domain could simplify the logical structure of your environment while keeping your DNS namespaces separate from Active Directory.

If your organization makes extensive use of its separate namespaces, you may want to consider a design like this. Each domain tree in the forest can then maintain a certain degree of autonomy, both perceived and real. Often, this type of design will seek to satisfy even the most paranoid of branch office administrators who demand complete control over their entire IT structure.

Real-World Design Example

To gain a greater understanding of the times an organization might use this particular design model, let's look at the following AD structure. City A is a local county governmental organization with a loose-knit network of semi-independent city offices such as the police and fire departments that are spread out around the city. Each department currently uses a DNS namespace for name resolution to all hosts and user accounts local to itself, which provides different e-mail addresses for users located in the fire department, police department, and other branches. The following namespaces are used within the city's infrastructure:

- citya.org
- firedeptcitya.org
- policeofcitya.org
- cityalibrary.org

The decision was made to merge the existing network environments into a single Active Directory forest that will accommodate the existing departmental namespaces but maintain a common schema and forest root. To accomplish this, Active Directory was established with citya.org as the namespace for the root domain. The additional domains were added to the forest as separate trees but with a shared schema, as shown in Figure 5.7.

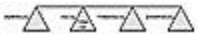


Figure 5.7 Single Active Directory forest with separate directory trees for departments.

The individual departments were able to maintain control over their individual security and are disallowed from making changes in domains outside their control. The common forest schema and global catalog helped to increase collaboration between the varying organizations and allow for a certain amount of central administration.

This type of domain design is logically a bit messier but technically carries the same functionality as any other single forest design model. All the domains are set up with two-way transitive trusts to the root domain and share a common schema and global catalog. The difference lies in the fact that they all utilize separate DNS namespaces, a fact that must also be reflected in the zones that exist on your DNS server.

Federated Forests Design Model

A new feature of Windows .NET Server 2003's Active Directory implementation is the addition of cross-forest transitive trusts. In essence, this allows you to establish transitive trusts between two forests with completely separate schemas that allow users between the forests to share information and to authenticate users.

The ability to perform cross-forest trusts and synchronization is not automatic, however, because the forest functionality of each forest must be brought up to Windows .NET functionality levels. What this means is that all domain controllers in each forest must first be upgraded to Windows .NET Server 2003 before any cross-forest trusts can be established. This can prove to be a difficult prospect for organizations already deployed with Windows 2000. Consequently, the federated forest design model is easier to consider if you do not currently have an Active Directory structure in place.

The federated forest design model is ideal for two different situations. One is to unite two disparate Active Directory structures in situations that arise from corporate acquisitions, mergers, and other forms of organizational restructuring. In these cases, two AD forests need to be linked to exchange information. For example, a corporate merger between two large organizations with fully populated Active Directory forests could take advantage of this capability and link their two environments, as shown in Figure 5.8, without the need for complex domain migration tools.

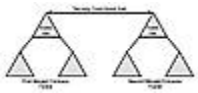


Figure 5.8 Cross-forest trust between two completely different organizations needing to share resources.

In this example, users in both forests now can access information in each other's forests through the two-way cross-forest trust set up between each forest's root.

The second type of scenario in which this form of forest design could be chosen is one in which absolute security and ownership of IT structure are required by different divisions or subsidiaries within an organization, but exchange of information is also required. For example, an aeronautics organization could set up two AD forests, one for the civilian branch of its operations and one for the military branch. This would effectively segregate the two environments, giving each department complete control over its environment. A one- or two-way cross-forest trust could then be set up to exchange and synchronize information between the two forests so as to facilitate communication exchange.

In Figure 5.9, a one-way cross-forest trust was set up between the civilian branch and the military branch of the sample aeronautics organization. In this example, this setup would allow only accounts from

the military branch to be trusted in the civilian branch, in essence giving the military branch users the ability to access files in both forests. As with NT domains, cross-forest trusts are one way by default. Unlike NT and 2000 trusts, however, cross-forest trusts in Windows .NET Server 2003 are transitive. To set up two-way transitive trusts, you must establish two one-way trusts between the two forest roots.

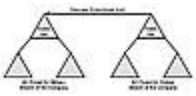


Figure 5.9 One-way cross-forest trust.

When to Choose Federated Forests

The concept of federated forests greatly enhances the abilities of Active Directory forests to exchange information with other environments. In addition, organizations that were reluctant to implement AD because of the lack of a solid security boundary between domains can now take heart in the ability of the federated forest design to allow specific departments or areas to have complete control over their own forests, while allowing for the transfer of information between the domains.

Real-World Design Example

To illustrate a good example of an organization that would choose a federated forest design model, let's consider fictional Conglomerate A, which is a food distributor with multiple sites worldwide. It currently operates a Windows .NET Server 2003 Active Directory implementation across its entire organization. All computers are members of the forest with a namespace of companyb.net. A root domain exists for conglomeratea.net, but it is not populated because all users exist in one of three subdomains: asia, europe, and na.

Conglomerate A has recently entered into a joint venture with Supplier A and would like to facilitate the sharing of information between the two companies and to exchange common address books. Supplier A also currently operates in a Windows .NET Active Directory environment and keeps all user and computer accounts in an Active Directory forest that is composed of two domains in the suppliera.com namespace and a separate tree with a DNS namespace of supplierabranch.org that reflects a certain function of one of its branches.

The decision was made to create a cross-forest trust between the two forests so that creden-tials from one forest are trusted by the other forest and information can be exchanged. The cross-forest trust was put into place between the root domains in each forest, as shown in Figure 5.10.

Figure 5.10 Cross-forest trust between root domains in each forest.

Remember, that just as in NT 4.0, a trust does not automatically grant any permissions in other domains or forests; it simply allows for resources to be implicitly shared. Administrators from the trusting domain still need to manually grant access. In our example, administrators in both forests can decide what resources will be shared and can configure their environment as such.

Source

http://www.informit.com/articles/article.aspx?p=32080&seqNum=8

Further Reading:

- http://www.doecirc.energy.gov/documents/MS_Active_Directory_Design_Guide.pdf
- Microsoft Windows 2003/2008 Unleased
- Mastering Windows 2003/2008 Server