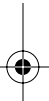
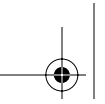


Design

part I

In this part:

- ◆ **Chapter 1: Active Directory Forest Design**
- ◆ **Chapter 2: Active Directory Domain Design**
- ◆ **Chapter 3: Domain Name System Design**
- ◆ **Chapter 4: Sites, Flexible Single-Master Operations and Global Catalog Design**
- ◆ **Chapter 5: Organizational Unit Design**
- ◆ **Chapter 6: Exchange Design Considerations**
- ◆ **Chapter 7: Hardware Sizing and Placement**



chapter I

Active Directory Forest Design

HOW DO YOU OPTIMALLY design a database that replicates only parts of itself to as many as thousands of domain controllers at differing intervals? Therein lies the need for this book. Active Directory may very well become the largest database implementation within your organization. If you talk to very many database administrators, you will see them cringe when you mention that you would like to replicate a database to multiple servers. But that is exactly what you are going to do with Active Directory and your domain controllers.

Throughout the first section of this book, I am going to discuss design criteria that you should consider when designing your Active Directory infrastructure. The chapters in Part I are organized simply. To do an Active Directory design, go through the chapters in order. By doing so, you will end up with a good understanding of the building blocks for a design that allows for efficient administration and control of the entire Active Directory environment.

Active Directory is a technology that is unlike most directory services in that it is an integral part of the operating system (OS). The many other directory services out there “sit on top” of the OS. Netware Directory Services (NDS), for example, can be easily upgraded independent of an OS upgrade. Not so for Active Directory. To update the directory service on the Microsoft platform, you currently need to upgrade to the latest Server OS or service pack. Each service pack brings a host of bug and security fixes for Active Directory, as well additional functionality; therefore, you should apply the latest service pack on all of your domain controllers within a month of its release—although that is easier said than done.

Take note that Active Directory enhancements, features, and functionality are greatly upgraded by Windows Server 2003. The move from Windows 2000 to 2003 requires very little planning. The two databases are built on the same underlying technologies, but Microsoft learned a lot after the initial rollout with Active Directory under Windows 2000. Consider Active Directory in Windows Server 2003 as version 2.0 to the 1.0 Active Directory version in Windows 2000—this is an unofficial versioning that’s included just for illustrative purposes. New administrative tools have been added and additional functionality has been included. However, for most of the added functionality, you will need to retire all of your Windows 2000 domain controllers. I’ll discuss more on that later in this chapter as we review the functional levels for domains and forests.

You should consider moving to Windows Server 2003 for a myriad of reasons, not the least of which is the support timeline expiration of Windows 2000. Mainstream product support expires five years after a product release. Many in the IT industry will take the conspiracy theorist's stance and accuse Microsoft of retiring products in order to keep companies on the purchasing side of the table. However, in our industry, you must admit that the technologies that develop over the course of five years tend to make operating systems and applications obsolete. For those of you who have left Windows NT 4 for the greener pastures of Windows 2000 and Active Directory, could you even imagine trying to perform some of the administrative tasks on Windows NT 4 that you can perform with the newer technologies?

TIP For more information on the lifecycle of operating systems and applications, peruse the article on Microsoft's website <http://support.microsoft.com/default.aspx?pr=lifecycle>

In the following sections, we are going to look at the criteria you should consider when developing your Active Directory design. Most of the information included comes from working with Active Directory over the past few years and the methodology that has proven to work the best. While some of the information may seem like it is common sense to you, there are times when common sense seems to take a back seat to the desire to implement technology.

Active Directory Forest Design Criteria

Active Directory design is both technically and operationally driven. It requires compromise among diverse groups that may be used to doing tasks their own way—DNS admins can use Unix for DNS, NetWare admins have a different architecture of directory services that they want to implement, ERP packages use their own directory service, proxy/firewall/internet access might use its own directory service, mail uses its own directory service.... You get the idea. How do you please everyone in your design? You probably won't. Start by developing the ideal design by yourself, if possible, and then let each group have its turn telling you what modifications they would like to see or, in a worst case scenario, why your design won't work. If you let each group try to design Active Directory, you'll never get done.

Get executive sponsorship in the design phase. I cannot stress this point enough. In other words, find an executive to approve the design phase of Active Directory with input from other groups (afterward). If you make an executive ally within the organization and they trust and like your plan, you will find that getting the design approved and moving on to the planning stages will be much easier. A college professor of project management once told me that executive sponsorship of projects is the number one indicator of whether a project will get done. He actually buys stock in companies that have good project management business processes. He says he does well with his stock picks.

Active Directory design is about putting structure around a chaos of unorganized objects. It's about administrative control and separating the service owners accountable for maintaining Active Directory and the services that support it from data owner administrators who are responsible for maintaining the objects within the directory. It is about architecting a solution that takes into account the limitations of the technology you are working with (Windows Server and Active Directory) and designing around the organizational day-to-day business. Your design needs to take into consideration speed/latency, name resolution, availability, security, disaster/recovery, hardware, etc.

Forest Design should be your first architectural element when designing Active Directory. A forest is the smallest instance of Active Directory. The forest is the topmost container in Active Directory. It is scalable beyond 5,000 domain controllers, 5,000 sites, and millions of users according to Microsoft's Branch Office Deployment Guide. Even the largest organizations should be able to contain all of the necessary objects within a single forest. You will find that other considerations will come into play when developing your design. Legislative, political, or organizational reasons may force you to move to a multiple forest design, but make sure there is a valid reason to do so. Later in this chapter, I will discuss the pros and cons of single and multiple forest implementations.

Although a forest is almost insanely easy to build, it is far, far more complex to design. Several options are available, and you need to know what roles forests and domains play within your organization. As you will see in the next section, the domain is no longer the security boundary, as it was under Windows NT 4. I will discuss the differences and the new technologies that make up the security boundary, replication boundary, administration boundary, schema, and Global Catalog.

Schema

A forest shares a single *schema*, which can be defined as the rules of what can go into a directory service. Active Directory is made up of *objects*, which are instances of an object class that have been defined by combining attributes to form what can be allowed within the directory. These rules also define where objects can be created and used within the directory service. Because all of the objects within the forest have to follow the same rules, there can be only one schema per forest.

Due to the important nature of the schema, you should not take its existence lightly. While you may not have to think about it on a daily basis, you will need to make sure that you do not allow anyone to have access to the schema. If changes are enacted within the schema, the results could be disastrous. Your organization may be one of the lucky ones that never have to modify their schema, but very few organizations are so lucky.

Many organizations will modify their default schema so that it will support directory-enabled applications. One such example, and probably the most popular, is the need to implement Exchange. Both Exchange 2000 Server and Exchange Server 2003 add many additional attributes and object classes to the schema. Prior to implementing an Active Directory-enabled application within your production environment, make sure you understand the ramifications of altering your schema. Test the application first in a test environment. Later, in the "Best Practices for Forest Design" section, I will discuss the need to change management. You should read this section if you want to control how changes are made to your infrastructure.

NOTE If you would like to see the schema extensions that are installed with Exchange 2000/2003, check out the information at http://msdn.microsoft.com/library/default.asp?url=/library/en-us/e2k3/e2k3/e2k3_1df_diff_ad_schema_intro.asp.

Schema Considerations

If you are extending the schema for an in-house application, consider contacting an Object Identifier (OID) issuing authority for the proper classification. Failure to do so could cause problems with other applications when they are installed within your environment. If an application needs to use an OID that is already in use, the application will fail the install. Windows Server 2003 will allow you

to reclassify an attribute; however, Windows 2000 will not. As a best practice, you should contact one of the following organizations to verify that the OID is not in use:

- ◆ Internet Assigned Numbers Authority (IANA) hands out OIDs for free under the “Private Enterprises” branch.
- ◆ American National Standards Institute (ANSI) hands out OIDs under the “US Organizations” branch for USD 1,000.
- ◆ British Standards Institute (BSI) hands out OIDs under the “UK Organizations” branch.
- ◆ Visit <http://www.iso.ch> for information on your country’s National Registration Authority.

While you are at it, register the OID so that no one else can use it for their commercial applications. Microsoft will validate all applications that you want to be certified for use within Active Directory. If you register your OID and someone else tries to use it, Microsoft will fail the application and the software vendor will have to change their application accordingly.

For information on registering OIDs for the ISO or if you are registering an OID under Microsoft’s branch, see the following websites:

- ◆ http://msdn.microsoft.com/library/en-us/ad/ad/obtaining_a_root_oid_from_an_iso_name_registration_authority.asp
- ◆ http://msdn.microsoft.com/library/en-us/ad/ad/obtaining_an_object_identifier_from_microsoft.asp

Security Boundary

The rules have changed since Windows NT 4. Under NT, the domain was the security boundary. If you were a member of the Domain Admins group, you had full control of your domain and were isolated from Domain Admins from other domains. Now with Active Directory, the forest is the security boundary in Active Directory (AD), not the domain. I get a kick out of clients who want to argue this point with me. Any Domain Admin on any domain controller throughout the forest can bring down the entire AD forest—either on purpose or by mistake. There are some simple ways to do this:

- ◆ Impersonate any user in the forest (in any domain).
- ◆ Read, change, or delete any Windows-secured resource or configuration setting on any machine (especially domain controllers) in the forest.
- ◆ Modify service accounts that run in the *system context*—that means operating system privileges.
- ◆ Run code in system context.
- ◆ Hide (bury) domain administrator–equivalent accounts for later use. I participated in an audit that found a hidden Admin account named GOD.
- ◆ Cause changes to replicate to other DCs. This is not a problem with database corruption; but it is for a denial-of-service (DoS) attack, which would be easy with a simple script that added users endlessly.
- ◆ Take ownership of files, folders, objects, attributes, and, thereby, breach privacy.

These are just a few of the many ways to bring down the Active Directory forest.

NOTE See Chapter 2, “Active Directory Domain Design,” for more information on what Active Directory domains are used for and the design drivers behind them.

When you are creating your Active Directory design, you must account for who will become the forest owner. The forest owner is any account that has full control access to every domain within the forest. Any domain administrator in the root domain of the forest (the first domain created in the forest) is automatically made a member of the Enterprise Admins and Schema Admins Groups in Windows 2000. You should be thinking like the rest of us and take users/Admins out of this group immediately.

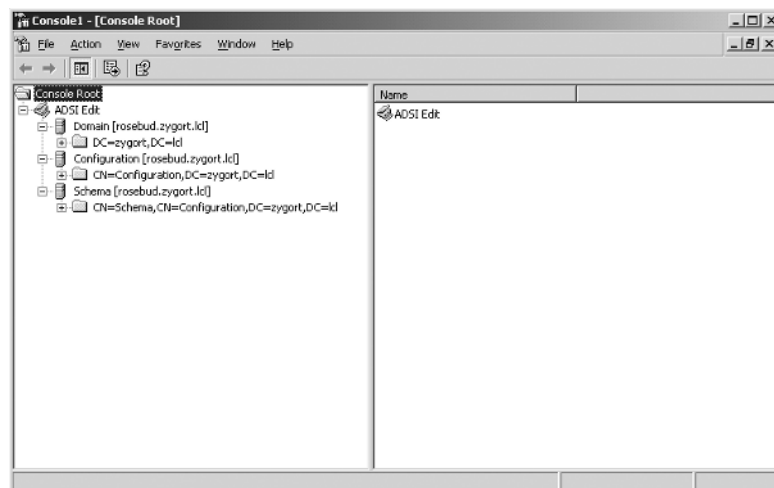
Replication Boundary

Active Directory forests provide for a complete replication boundary. Two Active Directory partitions, the configuration and schema partitions (or naming contexts), replicate on a forest-wide basis. Every domain controller within the forest will share identical data for these two partitions. Even if you have domain controllers from different forests within the same physical location, they will not share configuration and schema partition data; only those from the same forest will.

Another partition type, the application partition, can be configured to replicate to all domain controllers within the forest, but it is not mandatory that it do so. With an application partition, you have the ability to choose which domain controllers the partition will replicate to, thereby giving you a means of controlling some of the replication traffic within your organization. Later, in Chapter 3, “Domain Name System Design,” we will look at how the application partition works and the benefits of using this new partition type.

You can see the naming contexts upon opening Active Directory Services Interface (ADSI) Edit, which is provided in the Support Tools on the Windows 2000 (W2K) or Windows 2003 (W2K3) CD under \SUPPORT\TOOLS. Figure 1.1 shows ADSI Edit with the naming contexts opened.

FIGURE 1.1
ADSI Edit and the
Naming Contexts



As you will see in the following chapter, the domain partition, or domain-naming context, is replicated only to other domain controllers within the same domain. Although this does improve performance by restricting the amount of replication throughout the organization, it does cause issues when users are trying to locate objects within other domains. To alleviate some of the problems associated with partitioning the forest into separate domains, a Global Catalog was introduced.

A Common Global Catalog

A forest also provides for a common Global Catalog (GC) within the forest. A Global Catalog is a domain controller that hosts objects from every domain-naming context within the forest. At first you might think that could be a lot of data for a domain controller to host. If the Global Catalog server were to hold all of the attributes from every domain within the forest, you would be correct. However, to keep network traffic at a minimum, only around 200 of the 1,700+ available attributes for each object are copied into the GC. The GC is like a giant cache of directory objects and attributes that keep you from needing to query beyond a single domain controller. For example, you could easily take a laptop from domain to domain, country to country inside the same forest and authenticate immediately because your user object (and every user object in the forest) is cached in the Global Catalog, which replicates forest-wide.

You can control which attributes populate the Global Catalog with the Schema Management MMC snap-in. Be warned though, in Windows 2000, when you make a change to what values get put into the GC, you cause a full GC replication throughout the forest. This is not a big deal in an office of 200 people, but it is very much a concern when you have a 9GB GC such as Microsoft. This does not happen in W2K3; instead, only the changed or added attribute is replicated throughout the forest.

Later, as we determine the placement of domain controllers, we will come back to the Global Catalog server. There will be specific locations where you should place Global Catalog servers so that users and applications have access to the GC. For instance, the GC is Exchange 2000/2003's Global Address List (GAL), so you need to provide constant access to the GC for all of your Exchange servers.

If finding a resource easily across domain boundaries is important, without additional software services, consider a single forest. A single forest will allow all of the objects from every domain to be seen through the GC. If you must use multiple forests, there are applications, such as Microsoft Identity Integration Server 2003 (MIIS), formerly Microsoft Metadirectory Services (MMS), that can replicate objects and attributes between forests and differing directory services such as Novell's NDS to Active Directory.

TIP For more information concerning Microsoft Identity Integration Server 2003, visit Microsoft's website at <http://www.microsoft.com/miis>.

When you have multiple domains within your forest, you need to have a method to gain access to the resources in each domain. The Global Catalog servers need to have the ability to pull data from domain controllers in each of the other domains. In order to do so, trust relationships are used to determine which domains can contact each other.

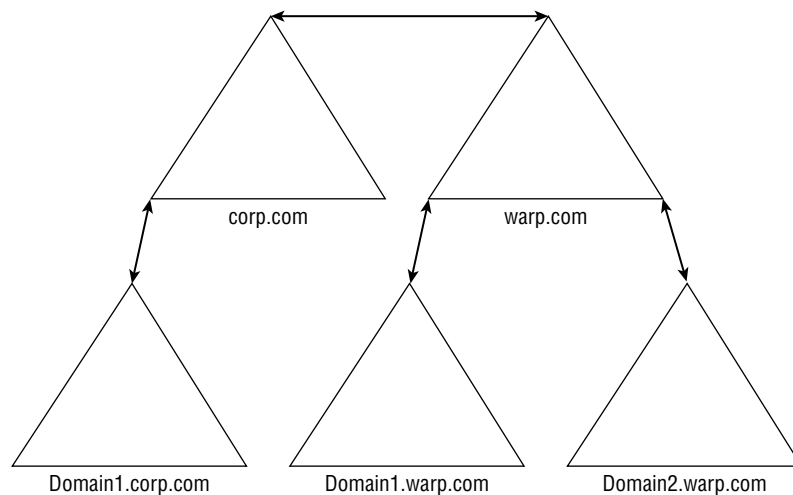
Kerberos and Trusts

Under the Windows NT 4 model, every domain was its own security boundary. To allow users to access resources within another NT domain, you had to create a trust relationship between the two domains. When you created a trust relationship, only one domain was allowed to trust users from the other domain. If you wanted to allow both sets of users to access each other's resources, two trust relationships needed to be created. To make matters worse, there was no sharing of trust. In other words, the trust relationships were not transitive. If DomainA had a trust relationship with DomainB, and DomainB had a trust relationship with DomainC, DomainA was still restricted from accessing DomainC until an explicit trust was set up between DomainA and DomainC. Needless to say, planning and maintaining the correct trust relationships in a large NT infrastructure caused many administrators to lose sleep.

Windows 2000 and Windows Server 2003 have changed the trust relationship game. Within a forest, all of the domains are interconnected through two-way transitive trusts. This allows every user within the forest to access resources within any domain within the forest so long as they have been granted permissions to access the resource. All of this is accomplished using the fewest trust relationships possible. Take a look at Figure 1.2. This is a typical forest that has two trees and several domains within each tree. In a Windows 200x forest, you will need only the trust relationships shown in the graphic. If you were to implement the same number of domains within a Windows NT 4 environment, you would need 20 trusts.

In order for the trust relationships to work within our forest, the Kerberos authentication service is used. With Kerberos, each of the domains and all of the security principles within each domain are identified and given access to the resources through a process known as *delegation*. As we look at domain design within Chapter 2, we will take a closer look at Kerberos and how it works across domain boundaries.

FIGURE 1.2
Active Directory trusts within a forest



Political and Administration Boundary

These two topics, political and administration boundaries, are tied together due to the fact that there is usually an underlying political reason that separate forests are created. As I mentioned before, the forest is the security boundary. Administrative accounts from the root domain of the forest have the ability to become the forest owners and, as such, can control all of the objects within the forest.

Although there are other reasons why you may want to create separate forests, doing so to appease a faction within your organization might end up being the deciding factor. Some divisions simply do not like allowing other administrators to be able to access their data. The only way that you can isolate their data is to create a separate forest and assign them as the forest owners.

The drawbacks to giving them their own forest are that you will have additional administrative overhead and you will need to make sure that they are properly trained to use Active Directory. If you can get by with a single forest, do so. You will reduce the total administrative costs. Later in this chapter, we are going to review the advantages and disadvantages associated with single and multiple forests.

The key to deciding where the administrative boundaries should be drawn will be dictated by whether you need to isolate services or data from other portions of the organization. Creating a separate forest is the only way you can isolate the directory services and data from groups of users. Before you make the decision to create a separate forest, review the arguments from all sides. You may be able to create a separate domain in order to give autonomy over services and data. Autonomy allows the administrators of certain resources to control them, while at the same time limiting the access to those resources from the other domains. Weigh the cost of a separate forest where isolation can be granted against the ease of administration with a single forest where you can apply autonomy. Of course, the political battles will ensue, as each division wants to have complete control over all of their resources. Don't forget what I mentioned earlier about executive sponsorship. If the battles rage on and you need to settle the disputes, there is nothing like having someone with lots of clout in your corner.

Multiple Forests Pros and Cons

Seeing multiple forests within a medium-sized client is not uncommon. One of my Active Directory clients had about 2,000 people and six forests. They used one for production, one for development, two for extranet applications, and two for development that mimicked the extranet production forests. This was a good, secure design for them. While not every organization will need to go to this extreme, I often recommend that you have a separate forest in which to test changes to Active Directory and software interaction. Creating a miniature version of your production environment will allow you to test changes before they are implemented within your production environment. Most companies that have a test environment have far fewer problems within their infrastructure than those that "shoot from the hip." I can't tell you how many times a service pack or hotfix has caused instability within a network.

I also like to recommend using a development forest if an organization has developers who need to test their software prior to implementing it within the production forest. Developers need their own forest if they require excess privileges or if they touch Active Directory. Often developers feel like they need Domain Admin access and a domain controller under their desk. I would never give developers this much power over my forest. As I mentioned within the "Schema" section, changes to the schema are not easily undone. Although Windows Server 2003 is a great deal friendlier when it comes to modifying the schema, you should never make any changes without first testing the implementation to

determine what the ramifications will be. I always recommend that developers do their work in a separate forest or, if possible, on virtual machine technology. Running a virtual system on an existing system is an easy way to mimic the production environment. The drawback is that the computer on which you are running the virtual system needs to have enough horsepower to run multiple operating systems at the same time. Two premier virtual system software applications are available. Microsoft sells their version, Microsoft VirtualPC (<http://www.microsoft.com/virtualpc>), which was originally marketed by Connectix, and EMC markets their own version called VMWare (<http://www.vmware.com>).

I also briefly mentioned a forest used for extranet applications. This is one area where you will need to determine the level of security you will need for users who access your infrastructure across the Internet. Some organization will implement a completely different forest for their perimeter network than they use within their internal network. This adds an additional layer of security to your design. If you were to use the same forest in both locations, you could run the risk of exposing information about your internal network if someone were able to hack into your perimeter network.

Figure 1.3 is a flowchart that will assist you in making decisions for your forest design. Within this flowchart, I take into account isolation and autonomy needs and choose the best forest design based upon the needs of the organization. In Table 1.1, you will find the advantages and disadvantages to using a single forest. Table 1.2 compares the multiple forest pros and cons.

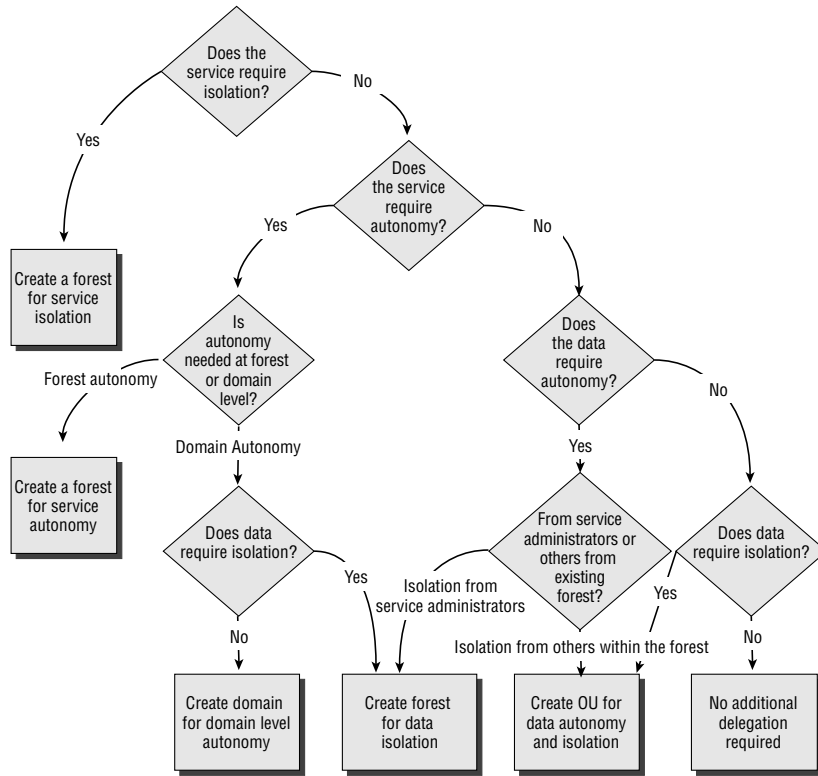
TABLE 1.1: SINGLE FOREST PROS AND CONS

SINGLE FOREST PROS	SINGLE FOREST CONS
Easier to administer.	Less secure for multiple business units with unknown/untrusted administrators.
Easier to troubleshoot.	Forests cannot be merged or split.
A single security boundary.	Domains cannot join other forests.
Single schema.	Schema differences are sometimes needed between business units.
Easier to support.	Cannot agree on change control within a domain. Users cannot search GCs of other forest without additional software.

TABLE 1.2: MULTIPLE FOREST PROS AND CONS

MULTIPLE FOREST PROS	MULTIPLE FOREST CONS
More secure.	More administration.
May have different schema in each forest (e.g., one business unit uses Exchange and another doesn't want its schema extended with the Exchange attributes).	Difficult to remember what schema extensions have been added and from which application they were added.
More control over outside trusts.	Don't have complete transitive trusts.
Trusts between forests in W2K3 are transitive Kerberos secured.	Trusts between two forests are one-way, nontransitive NTLMv2 in W2K. Certain Exchange 2000 mailbox features are not available when users exist in a different forest than their user object.

FIGURE 1.3
Flowchart to determine isolated or autonomous control



Most small and medium-sized companies will correctly opt for a single forest. Possible exceptions could be when:

- ◆ Extranet application(s) use Active Directory.
- ◆ Acquisitions and/or businesses break off into their own entities.
- ◆ Pilot deployments are needed to roll out application or new systems.
- ◆ Network administration is separated into autonomous groups that do not trust one another.
- ◆ Business units are politically separated.
- ◆ Business units must be separately maintained for legal reasons.
- ◆ There is a need to isolate the schema or configuration containers.

NOTE If part of your organization needs to have additional schema attributes and objects, and those attributes and objects are specific to an application, you could be able to use an Active Directory Application Mode (ADAM) partition, sometimes referred to as an application partition, instead of creating a separate forest. Note that if the objects are used for security purposes, they will have to be stored within Active Directory and not as an ADAM partition.

- ◆ There is a need to limit the scope of trusts between domains.

If you are considering a multiforest implementation, realize that you will need to do special planning for:

- ◆ DNS name resolution. Your solution is easy in W2K3 because you can use *conditional forwarding*, forwarding to preferred DNS servers depending on the domain name.
- ◆ Resource sharing. This includes things like printers, Global Catalog, certificate synchronization, trust relationships, and access control. The solution here is simply MIIS Server, a free download for W2K3 Enterprise Server users.
- ◆ Network infrastructure. It doesn't permit communication in a single forest (sites, subnets).

WARNING If you plan to use Active Directory in an outward-facing mode (extranet application), use a separate forest. If you need to synchronize accounts, download the free version of MIIS (formerly MMS) from <http://www.microsoft.com/miis>. You must be using W2K3 Enterprise Server to run it.

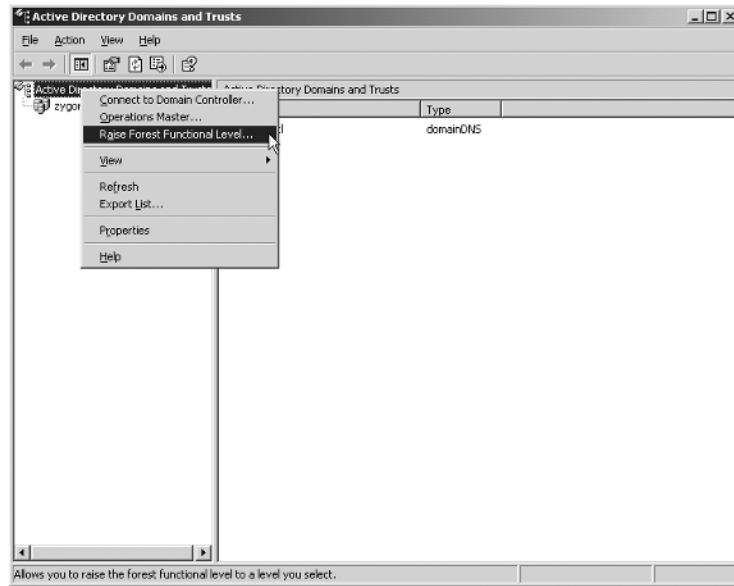
To get the most functionality from your Active Directory implementation, you will want to move to one of the higher *functional levels*. A functional level is a mode in which you can run Active Directory that allows it to utilize new tools that are not available if you have domain controllers from other operating systems within your infrastructure. Depending on what operating systems are running on your domain controllers, you will have the ability to change Active Directory's functional level to suit your needs.

Forest Functionality Mode Features in Windows 2003

Native mode in W2K is expanded into *forest functionality* and *domain functionality* in W2K3. Each of these two modes buys you extra AD functionality. Your goal in any migration, upgrade or clean install, is to get to *Forest Functionality level*, a term used with Windows Server 2003, as soon as possible. This is known as native mode in Windows 2000.

Your goal, in any upgrade, migration, or install, is to get to Windows 2000 native mode or Windows Server 2003 Forest Functionality mode as soon as you can. Forest Functionality means that you are no longer backward-compatible with Windows 2000 servers. In return, because all of the domain controllers are at their highest level, you can take advantage of all of the advanced feature sets of Windows Server 2003. Enterprise Admin rights are needed to upgrade to Forest Functionality level. As you can see in Figure 1.4, you will find the Forest Functionality choice in Active Directory Domains and Trusts. You can, however, also manually edit the functionality levels in ADSI Edit, and/or LDP.EXE, although I don't recommend doing so unless Microsoft support tells you to manually edit them.

FIGURE 1.4
Forest Functionality
level



When going to Windows Server 2003 Forest Functionality level, you gain the following features:

- ◆ Domain Rename, through the `netdom` command line utility.
- ◆ Link Value Replication. This is the ability, for example, for a group to replicate a single member when a change is made instead of the entire group being replicated each time (which is what Windows 2000 does). This also removes the 5,000-member limit on groups.
- ◆ The ability to convert a user in Active Directory to `INetOrgPerson` on-the-fly; plus gain the ability to put a user password on the `InetOrgPerson` LDAP object.
- ◆ Schema Redefine. This is not deletion, though it is the next best thing.
- ◆ Dynamic Auxiliary Classes. Use this when you need to have a departmental schema extension that doesn't affect the rest of the forest. For example, use it when a department needs to put employee IDs put into Active Directory and you, as the forest owner, don't want to extend the schema for this small department's needs.
- ◆ Basic and Query Group Types.
- ◆ Improved KCC (Knowledge Consistency Checker) algorithms and scalability. This feature is a big deal. In Windows 2000, you were told to turn off the KCC for implementations where you had about 100 DCs in a domain. Now, the KCC can scale to more than 4,000 DCs in a domain, although you'd probably never want to have a domain that big.
- ◆ Additional attributes automatically added to the Global Catalog.

- ◆ ISTG (Inter-site Topology Generator) ability to stop replicating with an “off-line” domain controller automatically.
- ◆ Constrained Delegation. This allows you to control the service principle names of the service accounts that another service account can be delegated.
- ◆ Application Groups. This is a method of controlling the accounts that have access to an application.
- ◆ Cross Forest Trusts. These are perfect for Exchange clients that exist in one forest, but have their Exchange mailbox in another forest. They eliminate the need to relogin. Windows 2000 has NTLMv2 cross-forest trusts. Windows Server 2003 uses Kerberos trusts. (Can you say single sign-on?)
- ◆ Update logon time stamp as a fast synching replicated attribute.
- ◆ Universal groups (same as in native mode for Windows 2000).
- ◆ Group Nesting (same as in native mode for Windows 2000).
- ◆ Switching distribution groups to security groups and vice versa (same as in native mode for Windows 2000).
- ◆ SID History as an attribute of a user object (same as in native mode for Windows 2000). This is a very important part of migrations that happen over time. An NT4 or Windows 2000 SID can be brought over during migration so that authentication against resources and objects in the NT4 or Windows 2000 domain/forest that have not been migrated will still work.

Best Practices for Forest Designs

As with any technical topic, you should follow best practices when you are designing. Active Directory is no different. You will want to make sure you follow each of the best practices described in the following sections in order to design an efficient and functional Active Directory forest.

Keeping It Simple: Start with a Single Forest

Review your current infrastructure and determine what works. Then review it again and look at what doesn't work. If you are moving from a Windows NT 4–based network, you will want to look at how you have your domains implemented and determine if you will need to create separate forests to isolate the data or consolidate them into a single forest. Simple designs are easier to maintain over the long run. Start with a single forest—one forest, one domain. Work from there to justify additional forests and domains. Having a single forest is like consolidating your environment; it saves money and time. You save duplication of administration efforts and expertise, thereby reducing your administrative overhead.

A single forest may contain several name spaces. Remember that name spaces are a function of domain design, not forest design; each name space requires a separate domain, but not necessarily a separate forest. Each of the name spaces will become its own tree within the forest. An organization that has multiple identities could utilize a single forest to facilitate sharing of resources. For example,

Microsoft is known for all of its diverse business ventures; `microsoft.com`, `msnbc.com`, `xbox.com`, and `msn.com` can all exist in the same forest.

Aiming for the Ideal Design

Aim for the absolute best design scenario based upon the requirements and priorities of your organization. Be open, though, to evaluate several alternatives. Once you have created what you determine to be an optimal design, let others within the organization review your design and allow them to have input. Not every design will fall neatly into specific categories, so be flexible. If you based your design on best practices, little structural change should need to be implemented. Also remember that the design process is always in a state of flux. Businesses change, sometimes on a daily basis.

Designing with Change Control Policies in Mind

This is an often overlooked planning step. Realize that extending the schema later (to implement or upgrade to Exchange, for instance) affects the entire forest, every object, and the domain controller. Have a Change Control policy in place that emphasizes proper operational processes. I know this is boring and a pain, but it is sorely needed. Everything you do should be verified in a lab first, monitored, rolled out to a pilot group, verified, and then deployed in a systematic approach. Anticipate change, reorganizations, politics, etc.

Before deciding to make any type of change, make sure you verify that the change that will be implemented is required. If you are simply adding an attribute to an object class, make sure none of the existing attributes will support your needs. There are 15 custom attributes that can be used for any purpose. Using these attributes will only cause replication traffic to the domain controllers in the domain where the object exists. If the change needs to be added to the Global Catalog, the change will replicate throughout the forest. Be aware of the additional replication you will be introducing.

If you do add an additional attribute or attributes to an object class, the new object class will need to replicate to all domain controllers within the forest. If you have WAN links where replication is controlled through a schedule, you could introduce inconsistencies between your domain controllers until the replication has completed. You will also cause additional replication across those WAN links, which could pose other problems for the users who are trying to use them. Make sure you know how you are going to roll out the changes and the schedule you will enforce.

A well-defined Change Control policy will reduce the problems associated with making changes to your infrastructure. Debates will rage among the administrators of the domains within the forest. The battles that need to be fought to prove the change needs to be put into place are hard enough without having to decide on all of the criteria to include within your policy. The criteria should include the following:

- ◆ Planning and testing. The planning documents that must be completed and the types of tests that the change will need to pass.
- ◆ Who is able to make the change. The appropriate parties that will be able to enact the change within the schema.
- ◆ The rollout schedule. Project guidelines for how the change will be made.
- ◆ Where the change can be made. The systems that will be used to change the schema.

Prior to implementing Active Directory, decide who will make up the approval committee. Meet with the administrators of all of the domains within the forest to gain approval of the policy. Everyone who will be affected needs to buy in. Getting them to do so may not be a simple task, especially in an environment where you have service or data autonomy. Some administrators may not have a great deal of trust in other administrators. As mentioned before, make sure you have the appropriate allies so that you can get approval from the highest-level stakeholder.

Separating Extranet Applications into Their Own Forest

I can't stress enough that security is too important to take administrative shortcuts. Don't allow an extranet application in your DMZ to use the same forest as your production users. Simple hacks (such as enumeration, denial-of-service [DoS], and elevated privileges attacks) against your extranet could cause catastrophic damage to your internal network Active Directory structure if the two are linked with a single forest.

If you are running Windows Server 2003 as the operating system upon which Active Directory is operating, you will have options, such as conditional forwarding, that will make implementing separate forests easier. Even if you are still running Windows 2000, you should take the time to design a secure foundation for your Active Directory infrastructure, even if it will take additional administration and time.

Building a Design Based on the Standard Forest Scenarios

There are three basic forest design scenarios: the organization-based forest, the resource forest, and the restricted-access forest. Each of them is used for specific design goals.

Organization-Based Forest Organization-based forests are the most common type of forest. Using this design, an organization's resources are placed within the forest and organized so that the appropriate groups have control over the resources they need. While companies will choose the decentralized model for several reasons, one of the primary reasons is autonomy of control. If autonomy is required, a department or division could have a domain created for them, or OUs could be built within a domain.

Resource Forest If certain resources need to be isolated from the rest of the organization, then a resource forest can be created. A resource forest is one in which the resources (such as shared folders, printers, member servers, etc.) are located within their own forest so that the owners of the resource can control them. Usually, the resource forest will not have any user accounts within the forest with the exception of the service administrator accounts that are required to maintain the forest. Using this type of forest, an organization can make sure that the resources contained within the resource forest are not affected by the loss of services in any other forest.

Restricted-Access Forest A restricted-access forest creates complete separation of service administrators. While trust relationships can be built to allow users to access the resources within the remote forest, the service administrators from the two forests will not be allowed to administer the other forest services. If there is any need for isolation of services, this is the type of forest structure that will need to be built.

ADDITIONAL RESOURCES

The following are white papers and websites that will help you familiarize yourself with some of the important Active Directory topics:

Multiple Forest Considerations Whitepaper <http://www.microsoft.com/downloads/details.aspx?displaylang=en&familyid=B717BFCD-6C1C-4AF6-8B2C-B604E60067BA>

Design Considerations for Delegation of Administration in Active Directory <http://www.microsoft.com/technet/prodtechnol/windows2000serv/technologies/activedirectory/plan/adde1adm.mspx>

Best Practice Active Directory Design for Managing Windows Networks <http://www.microsoft.com/technet/prodtechnol/windows2000serv/technologies/activedirectory/plan/bpaddsgn.mspx>

Active Directory Information <http://www.microsoft.com/ad> and <http://www.microsoft.com/technet/ad>

How to Create a Cross-Reference to an External Domain in Active Directory in W2K
<http://support.microsoft.com/?id=241737>

Next Up

You should have a good understanding of the elements that go into a forest design. However, before you can actually create the forest, you must make sure you have the appropriate domain design criteria in place. The forest is created as soon as the first domain is built. But before you jump into promoting a Windows 2000 or Windows Server 2003 system to a domain controller, you will need to make sure that the domain design for your forest is well thought out and will support your organization's needs. The next chapter will assist you in making good design decisions based upon the administrative and support needs of your organization.