ITPro™ SERIES

Windows & .NET MAGAZINE eBooks

Migrating to Windows Server 2003, Active Directory, and Exchange 2003

By Dan Holme and Dan Balter

Contents

Chapter 2

Active Directory Design

A solid Active Directory design is, of course, a prerequisite to migration. Without a well-thought-out structure of forests, trees, domains, organizational units (OUs), and sites, your migration will be a road to nowhere, fraught with failure. In this chapter, you will learn the ins and outs of Active Directory design. We assume that you are familiar with fundamental constructs such as domains and that you have had some Active Directory education or design experience. Therefore, our goal is to recast design considerations in a real-world perspective and present the experience we've had with good and bad designs. We also want to make sure to equip you with the latest knowledge regarding design. Too many resources are based on early Microsoft Windows 2000 deployments. Since that time there have been significant changes in both theory and practice.

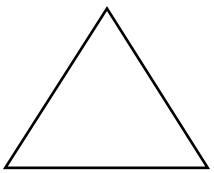
A Brief Overview of Key Active Directory Elements

Before we begin examining design considerations, let's briefly define key Active Directory structural elements: domains, trees, forests, and sites. A *domain* is the fundamental logical structural unit of Active Directory. You cannot have Active Directory without at least one domain and you cannot have a Windows 2000 or Windows Server 2003 domain without Active Directory.

Each domain requires at least one domain controller (DC) that hosts a copy of the domain's database. Unlike Windows NT, Active Directory DCs are not divided into primary and backup domain controllers. Each DC can write to the directory and replication takes place in a multimaster topology that ensures any change to the directory on any DC will replicate to all appropriate DCs.

When diagramming Active Directory, domains are represented by triangles as Figure 1 shows.

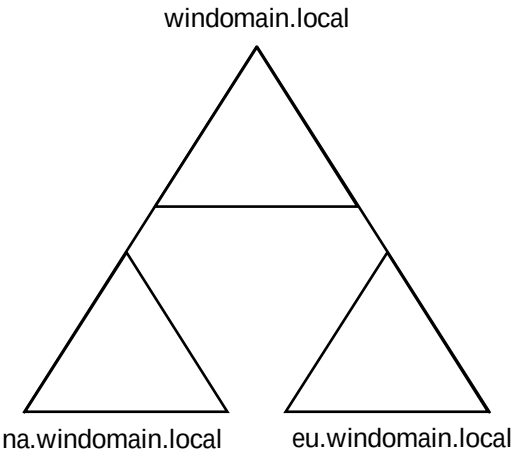
Figure 1:
Representing an Active Directory domain



Active Directory domain names follow DNS standards, which provide a hierarchy in the domain namespace of Active Directory. For example, the domain in Figure 1 might be called windomain.local.

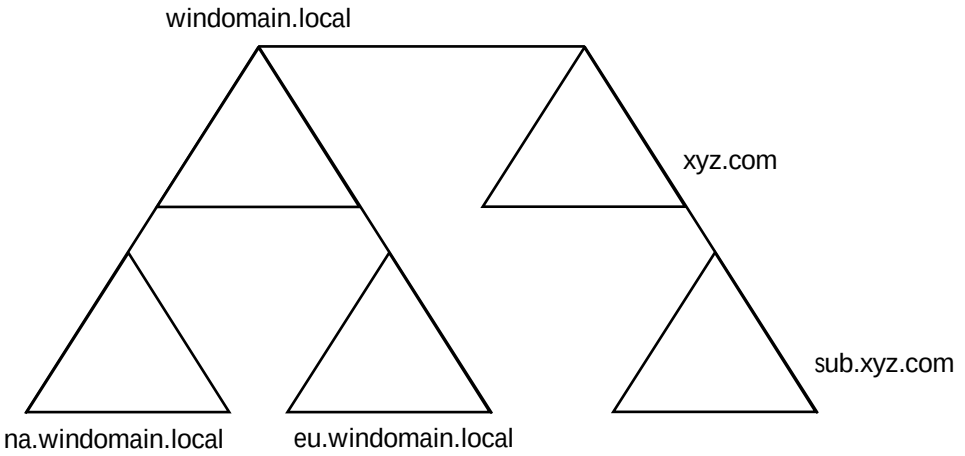
Figure 2 shows a *tree*, which is one or more domains in a contiguous DNS namespace (i.e., each domain shares a common root in the DNS namespace).

Figure 2:
Representing an Active Directory tree



A *forest*, which Figure 3 shows, is one or more trees that do not share a contiguous DNS namespace. A forest is the boundary of an Active Directory. A forest must have at least one domain (at least one tree) and can support many domains in many trees.

Figure 3:
Showing an Active Directory forest



The first DC installed in the forest creates the *forest root domain*. The forest root domain has important characteristics, as you will learn, so think carefully about which domain you will install first. The forest root domain is the first domain installed in an Active Directory enterprise regardless of its location in any DNS namespace hierarchy.

The final key structural component of Active Directory is the *site*. An Active Directory site represents a portion of your network topology that is highly connected. Active Directory sites will often parallel what you refer to as your enterprise's sites or locations.

Forest Design

Forest design discussions need to focus the relationship between specific forest and enterprise characteristics. Too often, forest design discussions lose focus and consider issues that have nothing to do with the true role of an Active Directory forest in an enterprise network. Careful evaluation degrades into a "kitchen sink" conversation and poor design decisions result. The tendency to consider less relevant and irrelevant issues is exacerbated by the plethora of outdated white papers and Active Directory design guidelines written by Microsoft and others during beta phases of Windows NT 5.0 (later, Windows 2000). Five years of experience with Active Directory and thousands of implementations have clarified the issues that are, and are not, relevant to an effective forest design.

Forest Characteristics

An Active Directory forest represents a single deployment of Active Directory. The forest is the boundary of Active Directory —a forest can have only one Active Directory and an Active Directory can have only one forest. An enterprise might have more than one forest, in which case it has multiple, completely distinct implementations of Active Directory.

A forest has five important characteristics:

- A single *schema* defines the attributes and object classes that can exist within the forest. The schema determines numerous other characteristics of the forest, including default security for object classes and attributes, indexing behavior, and global catalog (GC) contents.

Note

Members of the Schema Admins group, hosted in the forest root domain, can modify the schema. The default member of the Schema Admins group is the Administrator account in the forest root domain.

- A single *configuration* defines the domains, site topology, replication, and many of the services within the forest.
- A single *global catalog* contains information about every object in the forest. Because the GC includes an extensible subset of attributes for every object—specifically the attributes that are most often used to search for an object—it facilitates finding objects in the forest.
- *Two-way transitive Kerberos trust relationships* provide for authentication of any security principal (e.g., user, group, computer, inetOrgPerson) in the forest.
- An *Enterprise Admins group*, hosted in the forest root domain, is the default owner of all domains in the forest and therefore can correct errors anywhere in the directory.

Forest Design Tenets

The first question that you need to answer when designing your Active Directory is, "How many forests do I need?" Keep the following considerations clearly in mind as you evaluate forest design alternatives.

A single forest is the most straightforward design

Active Directory design guidelines commonly proclaim the single-forest design as the best practice. It is without doubt the most straightforward design with the lowest cost in terms of hardware and administrative overhead. The vast majority of Active Directory implementations are single-forest designs.

However, the knowledge we've gained during our last five years of experience and recent developments in the technology (including cross-forest trusts in Windows 2003 and metadirectory services such as Microsoft Metadirectory Services, or MMS) suggest that the single-forest design is not always the best practice. Being the most straightforward and lowest-overhead model does not make it the best model in every circumstance.

A forest shares common security enforcement

The schema defines the default security for each attribute and object class. Therefore schema owners can enforce compliance with certain security policies. For example, local laws might require that personal data, such as social security numbers, be inaccessible to users but a directory-aware payroll application might need access to social security numbers stored in the directory. In this situation the schema owner can configure the ACL of the social security number attribute to let only the payroll application access employee social security numbers. In this way, a forest can provide for *enforcement of certain security policies.*

However, the schema does not define many common security policies, such as password policies, group membership, and resource access. These policies belong to directory data owners—domain and OU administrators. So even within a single forest, you cannot easily enforce many common security-hardening policies. You must instead configure the policies and monitor them for compliance. Because most common security policies fall into this latter category, you need to consider, but not overestimate, security policy enforcement in forest design.

A forest indicates and requires ownership of forest data and services

Forest data includes the objects in the schema and configuration as well as the forest root domain and GC. Forest services include the administration and support of DCs in the forest root and of site topology, replication, and services.

Some businesses have decentralized IT environments without individuals or teams capable of supporting forest-level data and services. In these situations, ownership of forest data and services might be more effectively placed in business units, regions, or divisions that can support forest ownership.

A forest implies strong levels of trust between administrators of each domain within the forest

A single forest is characterized by an Enterprise Admins group, hosted in the forest root domain, that is the default owner of each domain in the forest, and therefore can correct errors anywhere in the forest. Although this group should be highly secure and Enterprise Admin credentials are rarely used,

members of this group can take ownership of and alter any data in the forest. Similarly, members of the Schema Admins group can modify the schema and therefore affect the forest in a myriad of ways. Domain and OU owners must trust the forest owner to secure and manage membership of Enterprise Admins and Schema Admins carefully.

Because the forest is the security boundary of Active Directory, anyone with physical access or administrative credentials to DCs anywhere in the forest can introduce, either accidentally or maliciously, elements that can circumvent the security and integrity of the *GC*, schema, or configuration, and thereby present significant risk to all domains within the forest.

Therefore domain owners within a forest need to trust each other just as much as they trust domain administrators in their own domain. Domain owners need to be certain that the administration of each domain is in compliance with security policies and procedures, and that all DCs are physically secure. Alternatively, domain owners must be willing to accept the risks associated with less secure administration and security practices.

A forest implies high levels of collaboration between administrators of each domain within the forest

Because the configuration, schema, *GC*, and some services (e.g., DNS) must support the entire forest, domain owners must collaborate to determine policies and procedures related to these components. For example, a business unit might need to install a directory-aware application to respond to its business drivers. That business unit will need to work carefully with the data owner of the forest because the forest owner owns the schema. Only members of the Schema Admins group (hosted in the forest root domain) can modify the schema on the schema operations master (typically, a DC in the forest root domain). In addition, any schema extensions that the application creates will apply to the entire forest, not just to the business unit, so procedures for evaluating, testing, piloting, and deploying schema extensions must be in place and those procedures must be acceptable to all domain owners within the forest. This typically dampens, to some extent, a business' ability to adapt Active Directory to its unique needs.

Similarly, when a business unit represented in Active Directory as a domain needs to add a child domain, the forest owner must add the child domain. The domains in a forest also share the *GC*. Therefore when one entity decides to add an attribute to the *GC*, the forest owner must add the attribute and the attribute needs to be acceptable to all domains in the forest.

Finally, DNS must support name resolution for all domains in the forest. You need to consider any other network services (i.e., Microsoft or third party) that relate to the forest. For example, DHCP and Microsoft Remote Installation Services (RIS) servers running on Windows Server 2003 or Windows 2000 Server need to be authorized by the forest owner before they begin to function.

A forest implies high levels of collaboration between users in various domains in the forest

The two-way transitive Kerberos trusts within a forest provide for authentication of any security principal (e.g., user, group, computer, inetOrgPerson) in any domain in the forest. These trusts facilitate access to resources such as files and printers. To further facilitate access to common resources, users in any domain in the forest are part of the Everyone group in Windows Server 2003 (Windows 2000's equivalent is the Authenticated Users group).

Evaluating Forest Design Factors

We advise that you begin forest design with the assumption that you will end up with a single forest design. As mentioned earlier, a single forest is the most straightforward, lowcost model. Deciding between a single and multiple forest design will depend on the relative weights of forest ownership and administrative complexity.

Forest service and data ownership

However, if the enterprise cannot identify an appropriate forest owner (individual or group) to support forest-level data and services, or if divisions in the enterprise cannot subscribe to the level of trust and collaboration that is implicit within a forest, then a single forest might not be the best design. Changes to the schema, configuration, and other forest-level data and services are rare, but when changes are necessary it is usually due to a significant business driver. An entity in the forest will likely be dependent upon other entities, including the forest owner, to respond to its needs for modifications to forest data and services. The forest owner needs clear policies and procedures to provide adequate levels of responsiveness to entities in the forest.

Administrative complexity

Do not underestimate the administrative complexity of multiple forest models, particularly when collaboration across forests requires authentication. For enterprises in which cross-forest collaboration is common, administrative overhead is significant.

Traditionally, when users collaborate between domains, those domains require trust relationships. Those trust relationships multiply quickly. A simple scenario in which users in three domains in Forest A collaborate with users in three domains in Forest B requires nine trust relationships.

Conversely, in environments in which cross-forest collaboration is uncommon, administrative overhead (beyond forest service and data ownership concerns) is minimal. Consider also the *type* of collaboration that occurs. Email, Web-based application, and online collaboration tools (e.g., Lotus Sametime, Microsoft NetMeeting, Windows SharePoint Services) are often independent of Active Directory and, therefore, independent of trust relationships. Collaboration that does not require authentication by Active Directory does not depend upon trust relationships and therefore does not affect administrative overhead.

Recent developments in technology reduce the complexity of multiple forest models.

Cross-forest trusts

Windows Server 2003 supports cross-forest trusts, which enable trusts be established between the forest root domains of two forests. The cross-forest trust enables authentication for security principals in all domains in both forests—a result similar to authentication between domains in a single forest. You can secure cross-forest trusts in several ways to refine the level and type of authentication that the domains in the two forests support.

Note

Forests at the Windows Server 2003 functional level support cross-forest trusts. DCs in all domains in both forests must be running Windows Server 2003.

Metadirectory services
Metadirectory services, including Microsoft Metadirectory Services (MMS), further facilitate multiple-forest models by enabling the synchronization of objects in forests. For example, you can replicate a user account or group to several forests and then give that account access to resources in each forest without requiring a trust.

Domain Design

As with forest design, it is important to focus on the characteristics of domains that truly drive design. It is also crucial to remove prejudice and experience with Windows NT 4.0 domains from the picture. Windows NT 4.0 domains were severely limited based on the size restriction (40 MB of data) of the Security Accounts Manager (SAM) database; by the single point of failure at the primary domain controller (PDC); and the inability to delegate administrative control, such as control over a subset of users or the a single task, such as resetting user passwords. Active Directory is not subject to these limitations.

Domain Characteristics

Domains within a forest play several roles based on what are often referred to as boundaries: administrative, data, authentication, user account security policy, and policy-based administration boundaries.

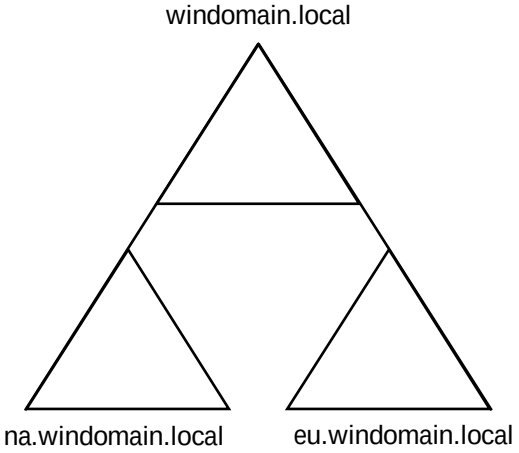
Administrative boundary

Domains serve as the administrative boundary for managing directory objects including users, groups, and computers. Within a domain, you can use OUs to create more granular administrative boundaries and to further delegate authority over objects and attributes.

The Administrators group of a domain is the owner of the domain, and therefore has extensive privileges and permissions to domain resources, including the directory. However, users who belong to the Administrators group in one domain do not, by default, have any privileges or permissions whatsoever in other domains in the forest. Note that this rule is true regardless of where a domain appears in the diagram of an Active Directory forest. For example, in Figure 4, the Administrators group of windomain.local does not have any authority over the na.windomain.local domain. The domain boundary is an administrative boundary, even though windomain.local appears to be a parent of na.windomain.local. The parent-child relationship is due to their DNS names and Kerberos trust hierarchy—that's all.

However, the administrators of windomain.local must have added the na.windomain.local domain to the forest because without the involvement of administrators in the forest root, domains cannot be added. When a child domain is added, the administrators of the parent domain are not granted any administrative authority. You need to add user accounts from the parent domain into appropriate groups in the child domain to achieve that type of administrative hierarchy.

Figure 4:
Illustrating Active Directory domain boundaries



Data boundary

Within a forest, domains act as a *partition* (i.e., division) of the Active Directory database. Each domain maintains all information about objects. The Domain NC stores all attributes for all objects in the domain. Using multimaster replication, the Domain NC replicates to each DC in the domain.

When more than one domain exists in a forest, the separate partitions for each domain ensure that data in one domain does not replicate to DCs in other domains. This characteristic is similar to Windows NT 4.0 domains.

Note

Remember, however, that the Global Catalog contains a partial attribute set for all objects in every domain in a forest, facilitating directory queries and object access.

Unlike Windows NT 4.0 domains, each Active Directory domain in a forest participates in the security unit of the forest, through the forest's trust relationships. In addition, as you learned earlier, every domain in a forest shares a common:

- Schema
- Configuration
- Global Catalog

Authentication boundary

A domain is responsible for performing authentication for a user account stored in its domain NC, even when that user is logging onto a computer in another domain. When a foreign user logs on to a domain, that domain can refer the authentication to the correct DC elsewhere in the enterprise's Active Directory or to trusted external domains.

User account security policy boundary

Domains in a forest also act as a boundary of directory related security policies. Administrators in the domain can specify policies such as password, DC security, account lockout, and Kerberos policies. The maximum scope of these policies is a domain. The policies that administrators specify in one domain do not, by default, flow into other domains in the forest.

Policy-based administration boundary

Group Policy enables you to centralize the administration of change and configuration management. Policies are implemented by linking group policy objects (GPOs) to sites, domains, and OUs. In a multi-domain environment, domains generally become the broadest scope for policy application.

Domain Models

Within a forest, the primary question you must answer is, "How many domains are required?" The only real difference between a tree and a forest is DNS namespace. Other than the default Kerberos trust partners, no functional difference exists between two domains that are part of the same tree or two separate trees in the forest. Therefore you do not yet need to consider the DNS hierarchy of the domain names. Your most important consideration is simply the number of necessary domains, including the justification for each domain.

Single domain model

In a single domain model, the forest root domain hosts all Active Directory objects—there is no other domain in the forest. The single domain model is the lowest total cost of ownership (TCO) implementation, and careful design of OUs lets the enterprise delegate and separate control of enterprise resources.

In Windows NT 4.0, you had to create multiple domains to achieve separation of control and delegation of administration. But by creating multiple domains, an organization sacrificed any ability to represent the needs and goals of the enterprise as an entity in its directory services. With Active Directory, the forest (even when it's a single domain) serves the needs of the enterprise and OUs serve the needs of units within that enterprise for various levels of autonomy and control.

Note

The single domain model provides the most effective and lowest-cost design for many organizations. It is a best practice for many enterprises, particularly smaller, simpler, or more centralized organizations.

Understanding and using the forest root domain

The first DC that you install in the forest creates the *forest root domain*. Note that the forest root domain is the first domain installed in an Active Directory enterprise regardless of its location in any DNS namespace hierarchy.

The forest root has roles and properties that set it apart from any other domains in the forest:

- The Domain Naming master, a role initially performed by the first DC you install in the forest root domain, manages the task of adding the first child domain or the first new tree in the forest.

- The forest root serves as the Kerberos trust *center* for all trees in the forest. When you add a new tree to the domain, the first domain in that tree (the new tree's root) establishes a trust with the forest root domain. The two-way trust lets the forest root domain and the new tree's root domain trust each other. Because all other trees' roots trust the forest root and because Kerberos trusts are transitive, the new tree root and its child domains are now in a trust relationship with all other domains in the forest.
- The Schema master, a role initially performed by the first DC you install in the forest root domain, manages the Active Directory Schema.
- The forest root domain hosts the Enterprise Admins and Schema Admins groups. These groups have ultra-powerful privileges and characteristics that affect the entire Active Directory enterprise.

A dedicated forest root domain

In multiple-domain models, you have the option of dedicating the root domain to its core roles of supporting forest-level data and services. Beyond the default and built-in objects, a dedicated forest root domain will not contain objects. In other words, you won't manage users, groups, or anything else in a forest root domain. You don't have a switch or checkbox to configure a dedicated forest root—what you do with the forest root makes it dedicated.

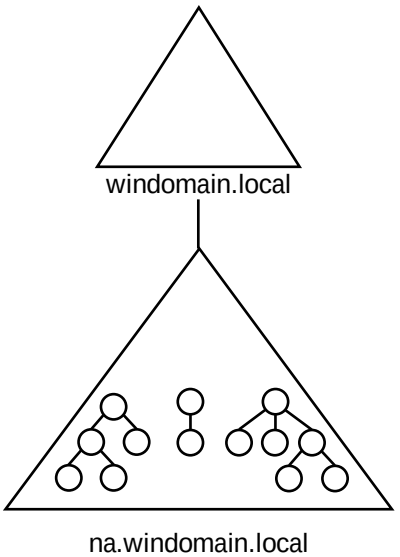
A dedicated forest root domain provides numerous advantages, including:

- Critical enterprise (i.e., forest) operations can remain in the forest root and can be isolated from other domains. These operations include the addition or deletion of other domains (by the Domain Naming Master) and the maintenance of the Schema (by the Schema Master). You will learn more about operations masters later in this chapter.
- The Enterprise Admins and Schema Admins groups enjoy the extra security afforded by their isolation from other domains. Only administrators of the forest root domain can modify the membership of these powerful groups.
- You can establish comprehensive auditing of the forest root without generating horribly unwieldy logs.
- You can establish the forest root domain in the Windows Server 2003 functional level, regardless of the existence of Windows 2000 DCs or Windows NT BDCs elsewhere in the Active Directory environment. We will examine functional levels later in this chapter.

Single global child domain model

In an Active Directory design, an organization might elect to dedicate the forest root by creating a second domain in the same tree (or a second tree in the forest) and by focusing all design decisions on the structure of the second domain, any additional domains, and the OUs within those domains. Figure 5 shows a single global child domain model with a dedicated forest root domain (windomain.local) and a single domain that maintains all accounts and objects.

Figure 5:
Representing a single global child domain model



Large or complex enterprises will find that a single global child domain model, with a dedicated forest root and a second *active* domain, provides an important enhancement to the stability and security of their enterprise. This stability and security provides pluses that outweigh the increased TCO of a multidomain model. This single global child domain model is a best design practice for many enterprises.

Multiple domain models

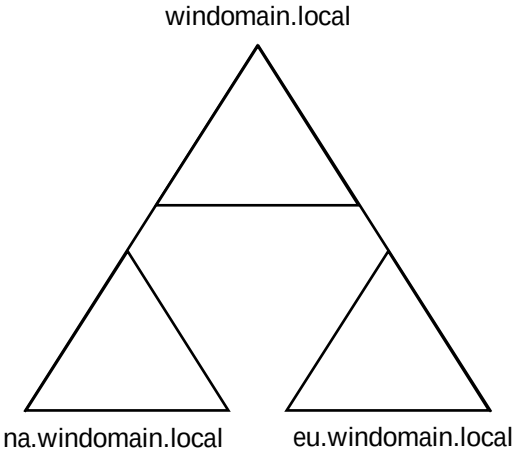
Multiple domain models consist of more than one domain in which you maintain regular directory data (e.g., users, groups).

Note

We highly recommend that if your organization will have more than one domain now or in the future that you develop a design with a dedicated forest root domain.

In Figure 6, windomain.local is the dedicated forest root domain. Two domains, na.windomain.local and eu.windomain.local, contain the actively supported directory objects.

Figure 6:
Representing two regionally located Active Directory domains



Evaluating Domain Models

In many—some would say most—enterprises, a single domain or a single global child domain model is the best practice design. There are a limited number of reasons why you would want more than one domain.

Divergent security policies

As we discussed earlier, you can configure certain security settings only at the domain level. Password policies (including length, complexity requirements, and how often they need to be changed) are an example of domain level security settings. You also have to configure Kerberos and lockout policies as well as trusted certificate authorities for the entire domain.

If your enterprise has divergent security requirements that cannot be reconciled, you will need to implement multiple domains to support that diversity. The need for divergent security requirements is a showstopper design consideration because user account policies are configured for all users in a domain.

Minimizing replication traffic

The most commonly cited justification for separate domains is the management of replication traffic. Because a domain is a partition (the Domain NC) in Active Directory, and because the Domain NC for a domain replicates only to DCs in that domain, you can reduce replication traffic with multiple domains. For example, in a two-domain model, attributes and objects in Domain A replicate only to DCs in Domain A and attributes and objects in Domain B replicate only to DCs in Domain B. Some organizations might consider creating domains for major regions, such as North America and Europe, as Figure 6 shows, to minimize replication over the slower and more expensive WAN links between regions.

Although this justification for separate domains is technically correct, you must evaluate it carefully given your specific environment. Replication traffic is minimal. Conservative best practice guidelines suggest that when the slowest link between two DCs in your environment is 28 Kbps, you need to limit the domain to 50,000 users. With a 56 Kbps or greater minimum link speed, the limit doubles to 100,000 users. Those guidelines applied to Windows 2000 and with improvements in replication for Windows Server 2003, the recommended limits will increase.

Replication will probably be one of the smallest consumers of network bandwidth in your enterprise. A typical email message is larger than most replication transactions. So although minimizing replication is a technically accurate justification for multiple domains, it is generally not a salient consideration in the real world.

Data isolation requirements

In some rare situations, an enterprise can be restricted from replicating certain data in the directory service. An example is an international bank with offices in Switzerland. If Active Directory stores the customer data, that institution has to isolate that data within a Swiss domain.

In the real world, data isolation requirements driving domain design is highly unusual. First, fine-tuning Active Directory to truly quarantine restricted data within a single domain in a forest is difficult because the GC replicates information about every object in the forest. Total data isolation occurs only in separate forests. Second, in situations with highly restricted data, Active Directory doesn't generally host the data. A separate isolated, secured and encrypted database would store the data.

Data autonomy requirements

Data autonomy relates to the idea of administrator realms of control in which administrators have full control of a realm and want to limit or prevent access by other administrators. Sometimes data autonomy reflects the reality of distributed, decentralized IT and other times it reflects politics.

In either situation, separate domains are rarely the answer. OUs provide data autonomy to the same extent as a domain. In addition to data autonomy, a domain denotes *service* autonomy and responsibility. A domain must have a coordinated user account security policy and must have ownership and support for Active Directory services (e.g., maintaining DC security, troubleshooting, disaster recovery). Rarely will a forest owner (for the enterprise) be willing to delegate authority over services in the forest solely to achieve data autonomy.

Remember that within a forest, administrators of all domains must have extraordinary levels of trust and confidence, as poor administrative practices in any domain have the potential for adversely affecting all domains in the forest. In other words, work through the politics by educating constituents as to the true nature of domains, forests, and OUs.

DNS Namespace Design

After determining how many domains are necessary to address your design drivers, you can plan the DNS names for the domains. Windows Server 2003 uses DNS as its preferred name resolution method, and Active Directory domains map directly to a DNS namespace.

When determining DNS names, be certain that they meet the following criteria:

1. *Uniqueness.* Each domain name must be unique, and not likely to change.
2. *Standards compliant.* Names should contain only standard characters (A-Z, a-z, 0-9, -).

3. *Secure*. Names should not be able to be resolved by external queries. To achieve this, use *split brain* DNS, in which a zone is maintained outside the firewall that contains host records for only those systems you want accessible by external clients. Internal DNS servers maintain a complete DNS zone with all records for hosts and services.
4. *Easy*. Make names that are easy to type and remember.

Typical DNS Designs

Many organizations choose to create a domain that is a subdomain of their existing namespace. Other organizations choose to use a .local namespace.

Subdomain

An enterprise might select to root their Active Directory in a subdomain of their existing namespace. For example, an enterprise with the registered domain name, windomain.com, might create a subdomain named *ad* (for Active Directory). Then its forest root domain becomes ad.windomain.com.

Although this approach ensures a unique, standard domain name for Active Directory, it poses one disadvantage: host names become rather lengthy and difficult to type and remember. For example, the name of a server in the single global child domain might be server01.na.ad.windomain.com. Organizations whose root Active Directory domain name lies deep in the DNS namespace might end up with incredibly long host names.

A .local domain

Alternatively, an enterprise might root its Active Directory in the .local namespace. Organizations can use the .local top-level domain name for internal name resolution without risk of interference or conflicts with other enterprises. The .local domain has no top-level DNS name servers, so external clients cannot resolve internal hosts on a .local namespace.

For example, an enterprise might create a forest root domain named windomain.local. This option ensures unique names that external hosts cannot resolve. This option also lets host names be higher in the namespace hierarchy. For example, a server in the North America domain might be named server01.na.windomain.local.

Interoperating with Existing DNS Infrastructures

Windows Server 2003 relies heavily on DNS and in an Active Directory domain every server, host, site, and service registers DNS records. Therefore, a rock-solid and dynamic DNS infrastructure is crucial to a successful implementation. An Active Directory domain can interoperate with third-party DNS, such as BIND. However, the best practice is to leverage Windows Server 2003's own DNS, which provides secure, dynamic DNS record registration and uses Active Directory's native replication mechanisms to provide incremental replication to targeted servers.

In an environment in which a DNS infrastructure already exists, you need to consider whether to integrate the Windows DNS domains into the infrastructure or to host them separately and link the two systems. By creating a DNS subdomain for Active Directory (e.g., ad.windomain.com) or using a unique namespace (e.g., windomain.local), you have the option of hosting DNS for your Active Directory domain on Windows Server 2003 DCs. By simply configuring forwarders on the DNS service, Windows clients can resolve names in the existing DNS namespace. By creating stub zones or secondary zones on the existing DNS servers, clients outside the Windows network can resolve

Windows hosts correctly. This approach lets an enterprise build an Active Directory domain and leverage the advantages of integrated DNS with very minimal effect on the existing DNS zones and servers.

NetBIOS Names

In addition to a unique DNS name, each Active Directory domain requires a NetBIOS name for compatibility with downlevel clients. For simplicity's sake, most organizations choose to use the DNS domain name (e.g., NA for North America) as the NetBIOS name.

UPN Suffixes

The *user principal name (UPN)* uniquely identifies a user within an Active Directory. A user's UPN must be unique to the Active Directory forest.

The UPN consists of a *name* and a *suffix* separated by the @ symbol (so it ends up looking like an email address). The UPN suffix is, by default, the DNS name of the domain. However, you can add any suffix you desire to the list of available UPN suffixes.

In multidomain models an enterprise might want to unify the namespace of UPN suffixes. For example, if your enterprise contains the domains windomain.local, na.windomain.local, and eu.windomain.local, you might want all users to log on as *username*@windomain.local.

To customize the list of UPN suffixes that are available to assign to a user account, follow the steps in the following procedure:

1. Open the Active Directory Domains and Trusts snap-in.
2. In the console tree right-click Active Directory Domains and Trusts, then select Properties.
3. Use the UPN Suffixes property sheet to add and remove suffixes.

OU Design

Within a domain, OUs are used to group objects that share common administration, configuration, or visibility. The meaning of this will become clearer as you learn more about OU design and management, but suffice it to say at this point that OUs provide an administrative hierarchy that accomplishes everything that individual Windows NT 4.0 domains did and much, much more.

An OU can contain millions of objects, but obviously it would not make sense to host every object in a single container. It is likely that objects are owned, administered, or configured in different ways, and that, at a minimum, you would want to collect objects to make them easier for administrators to find. You need to divide objects into logical, administrative containers.

As you do so, you will begin to create a hierarchy of OUs. A child OU, by default, inherits properties of its parent OU, including administrative delegation and Group Policy. Therefore, when you consider dividing one container into two containers, you must decide whether the new container needs to be a child of the existing container or a peer of (at the same level as) the existing container. Base your decision on whether the new container will have distinct properties from the existing container or whether the new container needs to inherit the properties of the existing container and have some unique properties as well. In the first case, the new container needs to be a peer at the same level. In the latter case, the new container needs to be a child of the existing container.

Creating an OU structure within an organization needs to follow a specific order and thought process:

1. Collect objects sharing common administration

OUs represent realms of data ownership or scopes of administrative responsibility. They are administrative containers used to facilitate the management of Active Directory objects. It is easier to perform many administrative and support tasks for users, groups, computers, and printers on a collection of objects rather than on individual objects. OUs are that collection and OUs can be nested to create a hierarchical structure of administrative containers.

You need to determine how to divide objects among a hierarchy of OUs. Start by imagining all of your users, groups, computers, and other directory objects in a single container. Then ask yourself, "How can I divide these objects based on how they are administered?" Identify how the objects should be grouped based on who owns or administers the objects. You are beginning the first phase of OU design: OUs should be structured, first, to reflect administration.

Let's say, for example, that your organization has regional IT administration centers, which support users in the eastern, central, and western divisions, and job tasks are divided such that the eastern administration center supports only eastern users and so on. Then you will probably want to divide the objects regional containers such as East, Central, and West. Each container—an OU—becomes a point of delegation. You can delegate levels of administrative authority over the objects in each regional OU to an appropriate group representing that region's administrators.

Continue dividing containers based on further administrative granularity. After this initial step of OU design, your OUs will reflect your enterprise's administrative model.

OUs need to provide sufficient data autonomy for administrators of business units, divisions, departments, and regions in most environments. Delegation of administration lets the enterprise grant either full control or any combination of granular permissions to an OU. In this way, OUs provide the administrative autonomy seen in Windows NT 4.0 domains without responsibility for domain and forest-level services. Therefore, as we discussed earlier in this chapter, you generally need to have more than one child domain when the enterprise requires unique user account security requirements.

2. Collect objects sharing similar configuration, application, or security settings

Because Group Policy lets you easily manage application, security, and configuration settings, you will want to collect objects that share similar configuration settings in a container or branch to which GPOs can be linked.

Therefore, in the second phase of OU subdivision, you need to divide objects that share similar configuration settings. Continuing the above example, let's assume you have engineers and accountants in the East regional OU. If the accountants will have a locked-down and tightly controlled user environment, but the engineers need more standard access and control of their systems, then you might want to divide the users in the East regional OU into Accounting and Engineering OUs. Then you can link a GPO that locks down the desktop to the Accounting OU without affecting the engineers.

Be wary of dividing OUs into too many levels. The best practice is to average only three to five levels deep. Group Policy is easiest to filter based on the OUs to which GPOs are linked. However, you can also filter GPOs based on group membership or Windows Management Instrumentation (WMI) information. So when you find yourself creating seventh-level and eighth-level OUs to link GPOs, you need to consider linking the GPOs to a higher-level OU and filtering the GPO. Then you won't require excessively deep OU structures. Deep OU structures affect object discovery and will

impede the performance of the directory. In addition, when you link GPOs to multiple levels of an OU tree, you increase the amount of time required for system startup and user log on.

Note

For more information about Group Policy design, implementation, and troubleshooting, see "Windows 2000 Group Policy, Profiles, and IntelliMirror," published by Sybex and written by Jeremy Moskowitz (http://www.sybex.com)

3. Collect objects for visibility

Administering objects within an OU that contains too many objects is difficult. One solution is to separate the objects in a large OU into smaller, child OUs. You can concentrate all administrative delegation and a Group Policy driven configuration on the parent OU. Child OUs and the objects they contain will inherit the configuration of the parent.

Note that this division for convenience and visibility is the last step in OU design. OU design needs to reflect administration, first; GPO requirements, second; and visibility, third.

Site Design

Whereas forests, domains, and OUs provide the logical structure to Active Directory, sites represent the physical topology of the enterprise. A site is an area of high connectivity, separated from other sites by slower links. Site design is independent of domain and OU design—a site can contain more than one domain, and a domain can contain more than one site.

The Functions of Sites

Sites support two major functions for the enterprise: service localization and control over replication.

Service localization refers to the ability of clients in a distributed environment to locate and use a service closest to the client such as authentication services that DCs provide. When a client logs on to an Active Directory domain, the client will attempt to use DCs in its site. Only when there are no DCs in its site will the client authenticate against remote DCs. Other services, including the GC and DFS, are *site aware,* which means clients will automatically use a local service when the service is available in both a local and a remote site.

Sites also drive the behavior of Active Directory replication. Within a site connectivity is good, so intrasite replication occurs frequently and uses uncompressed RPC over TCP/IP to ensure quick replication with minimal impact on the DC's processor. In a Windows Server 2003 site, a change to Active Directory replicates to all other DCs in the site within a few minutes. Between sites bandwidth can be at a premium, so replication is compressed and occurs less frequently. Administrators can configure the frequency and schedule of intersite replication.

The rules used to define sites vary widely from enterprise to enterprise. Microsoft's published best practices suggest that the minimum link speed within a site should be 512 Kbps. Any link slower than that should be used to define the boundary between two sites. In practice, some organizations have links as slow as 28 Kbps within a site.

The reason some organizations keep slow links within a site is that an Active Directory site should be used only to define service localization and replication, not to reflect what you call sites or

locations when you refer to your network topology. A slow link might connect a location in your network topology that is small and does not warrant having DCs or other services. Defining the small location as an Active Directory site would make no sense because the site would offer no benefits.

You need to examine each location and each link carefully. Generally, if a location doesn't warrant a DC, you are unlikely to define that location as an Active Directory site. This point will become clearer as we look at server placement later in this chapter.

So, when defining your sites, examine your network topology, locations, and link speeds. For each location that is connected using a slower link, consider whether the location warrants being defined as an Active Directory site based on whether it will support services—particularly DCs.

Server Placement

The services that are most crucial to Windows clients include authentication (by DCs), DNS, WINS, DHCP, and the GC. When considering a branch, an office, or a remote location, you need to be certain that these services are available to users in that location. Either the link needs to be sufficient and reliable or the services need to be hosted on local servers.

Domain Controllers

Before you start placing DCs in every remote site, consider the management and security implications. Additional DCs mean additional administrative overhead, and anyone with physical access to a DC can introduce problems into the domain. Some enterprises migrating to Windows Server 2003 are choosing to centralize the DCs in their environment, rather than continuing to support DCs in remote sites.

When you decide whether or not to place a DC in a remote site, consider where the greatest pain will lie if the link to the site goes down. Consider an organization in which a datacenter at the corporate headquarters hosts the user data and email. If the link between a remote site and the headquarters goes down, users at the remote site won't be able to access data or email. And in such a case the presence of a local DC will not mitigate the pain. However, if there are significant local resources (e.g., files, printers, applications), then a local DC will let users authenticate for those resources even when the link to the headquarters is unavailable.

In Windows 2000, it was important to configure a GC server on a DC in every site. In a multidomain environment in the Windows Server 2003 or Windows 2000 functional level, users cannot log on unless their universal group membership can be determined (members of the Administrators group are exempt). The GC stores universal group membership. So when a Windows client cannot contact a GC server, it will deny logon.

Windows Server 2003 DCs now support universal group membership caching. With the feature enabled, when a user logs on for the first time, a DC will contact a GC server and will cache the user's universal group membership. Therefore, if a GC is unavailable on a subsequent logon, the DC can authenticate the user successfully.

Operations Masters

In any replicated database, certain functions can be performed by only one replica. Active Directory is no exception: certain operations are the responsibility of only one DC in a domain or forest. The terms below refer to these operations:

- Operations Masters

- Single-Master Operations
- Flexible Single-Master Operation roles (FSMOs)
- Single-Master roles
- Operations tokens

Regardless of the term used, the idea is the same. One DC performs a function, and while it does, no other DC can perform that function.

Although this might sound like a rehash of the Windows NT 4.0 PDC concept, it is not. Single-Master Operations are characteristic of any replicated database, and Active Directory Single- Master Operations bear striking differences to Windows NT 4.0 PDCs. All DCs are capable of performing operations. The DC that performs an operation is the DC that holds the operation token. An operation token, and thus the role, can be transferred easily to another DC without a reboot. To reduce the risk of a single point of the failure, the operations tokens can be distributed among multiple DCs.

Forest-wide operations

Domain naming and schema maintenance are the two Single-Master Operations roles in an Active Directory forest:

Domain naming

The domain naming role is used to add and remove child domains in the forest. The domain naming master must be connected and accessible to successfully add or remove domains.

Schema maintenance

The schema maintenance role lets the DC hold the token to make changes to the forest's schema. All other DCs hold read-only replicas of the schema.

Domain operations

Each domain supports three Single-Master Operations roles:

Relative Identifier assignment

The Relative Identifier (RID) master plays an integral part in the generation of security identifiers (SIDs) for security principals (e.g., users, groups, computers). Because any DC can create accounts and therefore SIDs, a mechanism is necessary to ensure that SIDs are unique. Active Directory DCs generate SIDs by appending a unique RID to the domain SID. The domain's RID master allocates pools of unique RIDs to each DC to ensure the SIDs that each DC generates are unique to the domain.

Infrastructure master

In a multidomain environment, it is common for an object to reference objects in other domains. For example, a group can include users, groups, or computers from another domain in its membership. The group's domain might not always have access to a DC from the account's domain, or even to a GC, so Active Directory creates a phantom object to represent the account. The phantom object includes only the object's SID, distinguished name (DN), and globally unique identifier (GUID). If that object is moved or renamed in its domain, its GUID does not change, but its DN changes. If that

object is moved between domains, its SID also changes. The infrastructure master contacts a GC or a DC in the foreign domain to periodically (every 2 days by default) examine phantom objects. Then the infrastructure master updates the phantom objects' properties with any changes and replicates those changes to other DCs in the domain.

PDC Emulator

The PDC Emulator role supports multiple, crucial functions for a domain:

- **Facilitates replication to downlevel BDCs.** An Active Directory domain in mixed mode supports the existence of Windows NT 4.0 backup DCs. BDCs don't understand Active Directory and don't participate in multimaster replication. Instead, they expect to receive directory updates from the PDC. One Active Directory DC in a domain must act like the PDC for these BDCs. The DC with the PDC Emulator token assumes this role.
- **Acts as a PDC for downlevel clients and tools.** To perform certain tasks such as password changes, Windows NT 4.0 and Windows 9x clients without the Active Directory client contact the PDC of a Windows NT domain because only the PDC can write to the domain's directory. Downlevel clients are unaware that all Windows Server 2003 and Windows 2000 DCs can write to the directory. So the DC holding the PDC Emulator token registers itself and responds like a Windows NT 4.0 PDC. Downlevel tools, such as Windows NT's User Manager for Domains, and other Microsoft and third-party tools written for Windows NT domains, are also hard wired to connect to a PDC and will connect to the PDC Emulator.
- **Participates in special password update handling for the domain.** When a password is reset or changed, the DC making the change ignores all replication rules related to sites, site links, notification periods, and replication schedules, and replicates that change to the PDC Emulator. This special replication handling of password changes helps ensure that two DCs know about the new password as quickly as possible. When a user attempts to log on immediately after implementing a new password, the DC responding to the user's logon request might not know about the new password. But before it denies authentication, it authenticates against the PDC Emulator, which verifies the new password correctly and enables successful authentication.
- **Manages Group Policy updates within a domain.** Group Policy Editor attempts to bind to the PDC Emulator to avoid a situation in which a policy has been modified on two different DCs, thus creating a collision. Policy collisions cannot be resolved, and are almost certain to result in the loss of one or more policy updates. By binding to a specific DC, namely the PDC Emulator, the tool encourages administrators to focus on one source of policy information.
- **Manages time within a domain.** Active Directory, File Replication Service (FRS) and Kerberos rely on time stamps, so having all systems synchronized is crucial. The PDC Emulator in the forest root domain (the first domain installed in the forest) is the time master. The PDC Emulator in each domain in the forest synchronizes its time with the forest root PDC Emulator. The other DCs in a domain synchronize from the domain's PDC Emulator. All other domain members synchronize their time with their preferred DC. This hierarchical structure of synchronization, all implemented through the Win32Time service, ensures synchronization. Having the forest root domain's PDC Emulator synchronized with an external time server is crucial. If it is not, the event log will contain errors reflecting the situation. The Knowledge Base contains simple instructions for setting up the forest root domain's PDC Emulator to synchronize with an external time source.

- **Acts as the Domain Master Browser within the browse function.** A master browser in each network segment creates the browse list: the list of workgroups, domains, and servers. The domain master browser serves to merge the lists of each master browser so that browse clients can retrieve a comprehensive browse list.

Placement of Operations Masters

The best practice placement of operations masters is as follows:

Forest operations

The schema master and domain naming master roles should be placed on a single DC that is a GC server. These roles are rarely used and the DC hosting them should be tightly secured. The domain naming master should be hosted on a global catalog server, because the master must ensure that no other object of any type has the same name as a new domain being added. The GCs partial replica contains the name of every object in the forest.

Domain operations

The RID and PDC Emulator roles need to be hosted on only one DC. If the load mandates that the roles be placed on two DCs, those two systems should be physically well connected and have explicit connection objects created in Active Directory, so that they are direct replication partners. They should also be direct replication partners with standby RID and PDC Emulator DCs. In a mixed-mode domain, it is particularly important that the standby PDC Emulator is the same site as the active master.

The infrastructure master should be on a DC that is *not* a GC server, but should be physically well connected to a GC server. The infrastructure master should have explicit connection objects in Active Directory to that GC server so that they are direct replication partners. Note that if there is only one DC, or if every DC is a GC server, then it is OK to host the infrastructure master on a GC server. The infrastructure can be placed on the same server that acts as the RID and PDC Emulator.

Key Points

After a careful analysis of your enterprise, its network topology, and its administration, you can determine the following key components of your Active Directory design:

- **The number of forests.** The most straightforward and common practice is a single forest, although security concerns and service ownership concerns can result in an enterprise opting for a multiple-forest model. Due to cross-forest trust relationships and the availability of metadirectory tools such as MMS, supporting multiple-forest models in Windows Server 2003 is easier than in Windows 2000.
- **The number of domains.** The best practice and most common models are a single domain model and a dedicated forest root with a single, global child domain. In both of these models, the focus of all day-to-day administration is on one domain that contains the user, group, computer, and other objects representing the enterprise's resources, and ownership of those objects can be delegated using OUs. The dedicated forest root domain provides additional security for forest-level data and services, such as the schema and configuration.
- **The DNS namespace.** Active Directory domains need to follow DNS naming standards with common characters and short, easy-to-remember domain names. Using a subdomain of an

existing DNS namespace (e.g., ad.windomain.com) or a .local domain name (e.g., windomain.local) are common and effective.

- **The OU hierarchy.** Objects in Active Directory should be grouped first according to common administrative requirements, then according to change management (Group Policy) needs, and finally to make them easy to find.
- **Site topology.** Active Directory sites represent replication and service boundaries. Within a site, DCs replicate frequently. To conserve WAN link bandwidth, replication occurs less frequently between sites. In addition, clients using directory-aware services will use servers local to their site. By examining your network topology, including the location of services and bandwidth availability, you can design an effective site topology.
- **Server placement.** Ideally, crucial services such as DCs and name resolution services should be local to any site in which users need to access resources. In addition, you should plan for the placement of operations master roles including the domain naming, schema, RID, PDC, and infrastructure masters.