Microsoft

Technical Reference

# ACTIVE DIRECTORY
for Microsoft
WINDOWS SERVER 2003
Technical Reference

Mike Mulcare
Stan Reimer

PUBLISHED BY
Microsoft Press
A Division of Microsoft Corporation
One Microsoft Way
Redmond, Washington 98052-6399

Library of Congress Cataloging-in-Publication Data
Reimer, Stan, 1958-
Active Directory for Microsoft Windows Server 2003 Technical Reference / Stan Reimer, Mike Mulcare.
p. cm.
Includes index.
ISBN 0-7356-1577-2
1. Directory services (Computer network technology) 2. Microsoft software--Examinations--Study guides. 3. Microsoft Windows (Computer file) I. Mulcare, Mike II. Title.

TK5105.595.M85 2003
005.7'13769--dc21 2003042113

Printed and bound in the United States of America.

1 2 3 4 5 6 7 8 9 QWT 8 7 6 5 4 3

Distributed in Canada by H.B. Fenn and Company Ltd.

A CIP catalogue record for this book is available from the British Library.

Microsoft Press books are available through booksellers and distributors worldwide. For further information about international editions, contact your local Microsoft Corporation office or contact Microsoft Press International directly at fax (425) 936-7329. Visit our Web site at www.microsoft.com/mspress. Send comments to *mspinput@microsoft.com*.

Acquisitions Editor: Jeff Koch
Project Editor: Julie Miller

Body Part No. X08-73299

Contents

List of Tables xii
Dedications xiii
Acknowledgments xiv
Introduction xv

PART I Windows Server 2003 Active Directory Overview

1 **Active Directory Concepts 3**
- The Evolution of Microsoft Directory Services 3
  - LAN Manager for OS/2 and MS-DOS 4
  - Windows NT and SAM 4
  - Windows 2000 and Active Directory 6
  - Windows Server 2003 Domains and Active Directory 7
- Active Directory Open Standards 8
  - X.500 Hierarchies 8
  - Lightweight Directory Access Protocol (LDAP) 10
- Key Features and Benefits of Active Directory 12
  - Centralized Directory 12
  - Single Sign-On 12
  - Delegated Administration 12
  - Common Management Interface 13
  - Integrated Security 13
  - Scalability 13
- What's New in Windows Server 2003 Active Directory 14
  - Active Directory Users And Computers Improvements 14
  - Levels of Functionality 14
  - Domain Rename 15
  - Application Directory Partitions 15
  - Additional Domain Controller Installed from Backup Media 15
  - Deactivation of Schema Objects 16
  - Disabling Compression of Replication Traffic

    - Between Different Sites 16
    - Global Catalog Not Required for Logon 16
    - Group Membership Replication Improvements 16
    - Object Picker UI Improvements 17
    - Lingering Object Removal Mechanism 17
    - inetOrgPerson Support 17
- Summary 17

2 Active Directory Components 19

- Active Directory Physical Structure 19
    - The Directory Data Store 19
    - Domain Controllers 20
    - Global Catalog Servers 20
    - Operations Masters 23
    - Transferring Operations Master Roles 25
    - The Schema 26
- Active Directory Logical Structure 31
    - Active Directory Partitions 32
    - Domains 36
    - Domain Trees 37
    - Forests 38
    - Trusts 39
    - Sites 43
    - Organizational Units 46
- Summary 48

3 Active Directory and Domain Name System 49

- DNS Overview 49
    - Hierarchical Namespace 50
    - Distributed Database 51
    - Name Resolution Process 51
    - Resource Records 52
    - DNS Domains, Zones, and Servers 54
- DNS and Windows Server 2003 Active Directory 61
    - DNS Locator Service 61
    - Active Directory Integrated Zones 66
    - DNS Enhancements 69
- Summary 75

4 Active Directory Replication and Sites 77

- Active Directory Replication Model 77
- Replication Enhancements in Windows Server 2003 Active Directory 79
- Intrasite and Intersite Replication 80
  - Intrasite Replication 81
  - Intersite Replication 82
  - Replication Latency 83
  - Urgent Replication 83
- Replication Topology Generation 84
  - Knowledge Consistency Checker 84
  - Connection Objects 85
  - Intrasite Replication Topology 86
  - Global Catalog Replication 91
  - Intersite Replication Topology 93
- Replication Process 95
  - Update Types 96
  - Replicating Changes 96
- Configuring Intersite Replication 102
  - Creating Additional Sites 103
  - Site Links 103
  - Site Link Bridges 105
  - Replication Transport Protocols 106
  - Configuring Bridgehead Servers 107
- Monitoring and Troubleshooting Replication 108
- Summary 110

PART II Implementing Windows Server 2003 Active Directory

5 Designing the Active Directory Structure 113

- Designing the Forest Structure 113
  - Forests and Active Directory Design 114
  - Single or Multiple Forests 116
  - Defining Forest Ownership 119
  - Forest Change Control Policies 120

Designing the Domain Structure 121
Domains and Active Directory Design 121
Determining the Number of Domains 121
Designing the Forest Root Domain 124
Designing Domain Hierarchies 125
Domain Trees and Trusts 128
Changing the Domain Hierarchy 129
Defining Domain Ownership 130
Designing the DNS Infrastructure 131
Examining the Existing DNS Infrastructure 131
Namespace Design 132
Designing the Organizational Unit Structure 143
Organizational Units and Active Directory Design 143
Designing an OU Structure 144
Creating an OU Design 146
Designing the Site Topology 149
Sites and Active Directory Design 149
Networking Infrastructure and Site Design 150
Creating a Site Design 150
Designing Server Locations 153
Summary 158

6 **Installing Active Directory 159**
Prerequisites for Installing Active Directory 159
Hard Disk 160
Network Connectivity 160
DNS 161
Administrative Permissions 163
Active Directory Installation Options 163
Configure Your Server Wizard 163
Active Directory Installation Wizard (Dcpromo.exe) 164
Unattended Installation 165
Using the Configure Your Server Wizard 165
Using the Active Directory Installation Wizard 167
Operating System Compatibility 168
Domain and Domain Controller Types 169
Naming the Domain 171

    File Locations 172
    Verify or Install a DNS Server 173
    Selecting Default Permissions for User and Group Objects 175
    Completing the Installation 176
  Performing an Unattended Installation 178
  Installing Active Directory from Restored Backup Files 179
  Removing Active Directory 180
    Removing Additional Domain Controllers 182
    Removing the Last Domain Controller 183
    Unattended Removal of Active Directory 184
  Summary 184

7 Migrating to Active Directory 185
  Migration Paths 186
    The Domain Upgrade Migration Path 187
    The Domain Restructure Migration Path 189
    The Upgrade-Then-Restructure Migration Path 191
  Determining Your Migration Path 192
    Migration Path Decision Criteria 192
    Choosing the Domain Upgrade Path 193
    Choosing the Domain Restructure Path 195
    Choosing the Upgrade-Then-Restructure Path 197
  Preparing for Migration to Active Directory 198
    Planning the Migration 198
    Testing the Migration Plan 204
    Conducting a Pilot Migration 204
  Upgrading the Domain 205
    Upgrading from Windows NT Server 4 205
    Upgrading from Windows 2000 Server 213
  Restructuring the Domain 215
    Creating the Pristine Forest 217
    Migrating Account Domains 222
    Migrating Resource Domains 226
  Upgrading then Restructuring 231
  Configuring Interforest Trusts 232
  Summary 236

Part III Administering Windows Server 2003 Active Directory

8 **Active Directory Security 239**

- Active Directory Security Basics 239
  - Security Principals 240
  - Access Control Lists 240
  - Access Tokens 241
  - Authentication 241
  - Authorization 242
- Kerberos Security 242
  - Introduction to Kerberos 243
  - Kerberos Authentication 245
  - Delegation of Authentication 251
  - Configuring Kerberos in Windows Server 2003 253
  - Integration with Public Key Infrastructure 254
  - Integration with Smart Cards 257
  - Interoperability with Other Kerberos Systems 258
- NTLM Security 260
- Summary 260

9 **Delegating the Administration of Active Directory 261**

- Active Directory Object Permissions 261
  - Standard Permissions 262
  - Special Permissions 264
  - Permissions Inheritance 268
  - Effective Permissions 270
  - Ownership of Active Directory Objects 273
- Auditing the Use of Administrative Permissions 274
- Delegating Administrative Tasks 276
- Customized Tools for Delegated Administration 280
  - Customizing the Microsoft Management Console 280
  - Creating a Taskpad for Administration 281
- Planning for the Delegation of Administration 282
- Summary 283

10 Managing Active Directory Objects 285

Managing Users 285

User Objects 285

inetOrgPerson Objects 290

Contact Accounts 291

Managing Groups 292

Group Types 292

Group Scope 293

Creating a Security Group Design 296

Managing Computers 299

Managing *Printer* Objects 301

Publishing Printers in Active Directory 301

Managing Published Shared Folders 304

Windows Server 2003 Active Directory Administration Enhancements 305

Summary 306

11 Introduction to Group Policies 307

Group Policy Overview 308

Implementing Group Policies 311

Creating GPOs 312

Administering Group Policy Objects 313

Group Policy Inheritance and Application 314

Modifying the Default Application of Group Policies 316

Group Policy Processing 321

Delegating Administration of GPOs 326

Implementing Group Policies Between Domains and Forests 327

Group Policy Management Tools 328

RSoP Tool 328

GPResult 329

GPUpdate 330

Group Policy Management Console 330

Group Policy Design 332

Summary 333

12 Using Group Policies to Manage Software 335

Windows Installer Technology 336
  Creating a .msi file 336
Deploying Software Using Group Policies 337
  Deploying Applications 338
  Using Group Policies to Distribute Non–Windows Installer Applications 341
Configuring Software Package Properties 343
  Setting the Default Software Installation Properties 345
  Installing Customized Software Packages 345
  Updating an Existing Software Package 347
  Managing Software Categories 349
  Configuring File Extension Activation 350
  Removing Software Using Group Policies 351
Using Group Policies to Configure Windows Installer 352
Planning for Software Distribution Using Group Policies 354
Limitations to Using Group Policies to Manage Software 357
Summary 359

13 Using Group Policies to Manage Computers 361

Desktop Management Using Group Policies 362
Managing User Data and Profile Settings 364
  Managing User Profiles 364
  Folder Redirection 368
Configuring Security Settings with Group Policies 372
  Configuring Domain-Level Security Policies 372
  Configuring Other Security Settings 377
  Software Restriction Policies 379
  Security Templates 382
Administrative Templates 385
Using Scripts to Manage the User Environment 389
Summary 391

PART IV Maintaining Windows Server 2003 Active Directory

14 **Monitoring and Maintaining Active Directory 395**

- Monitoring Active Directory 395
  - Why Monitor Active Directory? 396
  - How to Monitor Active Directory 398
  - What to Monitor 410
- Active Directory Database Maintenance 411
  - Garbage Collection 411
  - Online Defragmentation 413
  - Offline Defragmentation of the Active Directory Database 414
  - Managing the Active Directory Database Using Ntdsutil 415
- Summary 417

15 **Disaster Recovery 419**

- Planning for a Disaster 419
- Active Directory Data Storage 420
- Backing Up Active Directory 423
- Restoring Active Directory 424
  - Restoring Active Directory by Creating a New Domain Controller 425
  - Performing a Nonauthoritative Restore 429
  - Performing an Authoritative Restore 431
  - Restoring Sysvol Information 433
  - Restoring Operations Masters and Global Catalog Servers 435
- Summary 440

Index 441

Tables

2 2-1. Domain Functional Levels 22
2-2. Forest Functional Levels 22
2-3. Group Policy Setting Types 48

3 3-1. Common Resource Records in Windows Server 2003 DNS 53
3-2. The SRV Record Components 62
3-3. A Subset of the DsGetDcName Flag Parameter Values 65

4 4-1. Replication Rings in a Complex Site 89

5 5-1. Linking Network Bandwidth to Site Link Costs 151

6 6-1. Enabling Client OSs to Log On to Active Directory 168

9 9-1. Special Permissions Configuration Columns 264

10 10-1. Account Properties for a *User* Object 288
10-2. Username Uniqueness Requirements 289
10-3. Active Directory Group Scopes 294

11 11-1. Group Policy Options 308
11-2. The Contents of the Group Policy Template 310
11-3. Configuring GPO Settings 314
11-4. GPMC Configuration Options 331

12 12-1. Deployment Options for a Software Package 343
12-2. Group Policy Setting Options for Windows Installer 353

13 13-1. Top-Level Containers in Default Domain Policy 363
13-2. Configuring User Profiles Using the Group Policy Object Editor 366
13-3. Password Policies 373
13-4. Account Lockout Policies 374
13-5. Kerberos Policies 375
13-6. Security Settings in Group Policies 378
13-7. An Administrative Templates Sampler 386
13-8. Default Templates Loaded in Windows Server 2003 388
13-9. Components of a Template Option 389

14 14-1. Core Active Directory Functions and Services 400
14-2. Replication Performance Counters 401
14-3. Key Security Volumes 402
14-4. Core Operating System Indicators 402

Chapter 2

Active Directory Components

Microsoft Windows Server 2003 Active Directory directory service exists on two levels: the physical and the logical. In terms of its physical structure, Active Directory is a single file on the server's hard disk and the hard disk of each of the domain controllers that host the service. The logical structure of Active Directory consists of the containers that are used to store the directory service objects (such as directory partitions, domains, and forests) in the enterprise. These directory partitions, domains, and forests ultimately are reduced to bytes that are stored in the physical components of the directory service. In this chapter, you will learn about the physical manifestation of the Active Directory directory service. Then, you will look at the logical structure of an Active Directory implementation. A solid understanding of the physical structure of the directory service is important, but your understanding of the logical structure is vital to successful implementation and management of your directory service infrastructure. It is this logical structure of the directory service that you will interact with on a daily basis.

Active Directory Physical Structure

The physical manifestation of Active Directory consists primarily of a single data file located on every domain controller in the domain. The physical implementation of Active Directory is described by the location of the domain controllers on which the service is hosted. In implementing Active Directory you can add as many domain controllers as are needed to support the directory services needs of the organization. There are five specific roles that each of these domain controllers can serve. These roles are known as *operations master roles*. Another role that any single domain controller in the domain can serve is that of a global catalog (GC). In this section you will look at both the Active Directory data store and the domain controllers that host it.

The Directory Data Store

All the data in an Active Directory database is stored in a single file on the domain controller—Ntds.dit. This data file is stored by default in the %SystemRoot%\NTDS folder on the domain controller. This file stores all the directory information for the domain, as well as some information that is shared by all domain controllers in a given organization.

A second copy of the Ntds.dit file can also be found in the %SystemRoot%\System32 folder structure. This version of the file is the distribution copy (default copy) of the directory database, and is used to install Active Directory. This file is copied to the server when Microsoft Windows Server 2003 is installed so that the server can be promoted to a domain controller without having to access the installation media. When the Active Directory Installation Wizard (Dcpromo.exe) is run, the Ntds.dit file is copied from the System32 folder to the NTDS folder. The copy stored in the NTDS folder then becomes the live copy of the directory data store. If this is not the first domain controller in the domain, this file will be updated from other domain controllers in the domain through the replication process.

Domain Controllers

By definition, any computer running Windows Server 2003 that maintains a copy of the Active Directory database is a *domain controller*. With several exceptions, which are detailed later in this chapter, all domain controllers are created equal. Using the multimaster replication process described in Chapter 4, "Active Directory Replication and Sites," every domain controller in the domain maintains an up-to-date copy of the domain database and is capable of making changes to the database.

In addition to the domain controllers that host Active Directory, there are several special-purpose domain controllers that Active Directory requires to perform certain functions. These are the *global catalog* (GC) servers and the *operations masters*.

Global Catalog Servers

A global catalog server is used to host the global catalog. The GC is a partial, read-only copy of all the domain naming contexts (NCs) in a forest. The GC contains a base set of attributes for every object in the forest (in every domain NC), but it is not a complete set. The GC data is derived from all domain directory partitions in the forest. The data is replicated to every GC using the normal Active Directory replication process.

Tip Whether or not an attribute is replicated to the GC is determined by the schema. Administrators can configure additional attributes to be replicated to the GC by using the Active Directory Schema snap-in on the Microsoft Management Console (MMC). To add an attribute to the GC, select the Replicate This Attribute To The Global Catalog option on the attribute itself. This will set the value of the *isMemberOfPartialAttributeSet* parameter on the attribute to *true*. You may choose to add an attribute to the GC if you anticipate that users will need to search for an object across the forest. Infrequently referenced attributes are not typically added to the GC.

The first domain controller installed in the domain is automatically a GC. Additional domain controllers can be designated as GCs by selecting the Global Catalog Server option in the Active Directory Sites And Services administrative tool. You may choose to designate additional domain controllers as GCs to optimize the logon process. How GCs are used by the logon process is described later in this section. Chapter 5, “Designing the Active Directory Structure,” provides more detail about the number of GC servers you will need to deploy and where to locate them.

You may wonder why GC servers are needed at all. One reason is that they are used for searching Active Directory. Without a GC, search requests received by a domain controller that did not possess the sought-after object would result in that domain controller referring the query to another domain’s domain controller. Because the GC contains a complete list of every object in the forest (though not every attribute of each object), the GC server can respond to any query using an attribute that has been replicated to the GC without needing to refer to another domain controller. The query that is sent to a GC server is a Lightweight Directory Access Protocol (LDAP) query using port 3268 (the default GC port).

The second, more complicated, reason for configuring GC servers is to process user logons. Ordinarily, every time a user logs on to a domain a GC is contacted. This is because non-GC domain controllers do not contain any information about universal group membership. (Universal groups are only available in domains with a functional level of Microsoft Windows 2000 native or Windows Server 2003. Functional levels are used in Windows Server 2003 to enable Active Directory features for all domain controllers that can support them.) Universal groups can contain user and group accounts from any domain in a particular forest. Since universal group membership is forest wide, group membership can only be resolved by a domain controller that has forest-wide directory information—the GC. In order for an accurate security token to be generated for the user seeking authentication, the GC must be contacted to determine the user’s universal group membership.

Note Windows Server 2003 supports a new feature known as *universal group membership caching* that makes it possible to log on to a Windows Server 2003 network without contacting a GC. Universal group membership is cached on non-GC domain controllers once this option is enabled and a user attempts to log on for the first time. Once this information is obtained from a GC, it is cached on the domain controller for the site indefinitely and is periodically updated (by default every 8 hours). Enabling this feature results in faster logon times for users in remote sites, as the authenticating domain controllers do not have to access a GC. To enable the universal group membership option in a site, open the Active Directory Sites And Services snap-in and select the desired site in the console tree. In the details pane, right-click NTDS Site Settings and then click Properties. On the Properties sheet, select the Enable Universal Group Membership Caching option and then select a site from which this site will refresh its cache. The <Default> option will refresh the site from the nearest site that has a GC.

Functional Levels

In Windows Server 2003, each forest, and each domain within a forest, can be assigned a specific functional level. These functional levels are used to enable features that are compatible on the combinations of operating systems supported by the various functional levels. When a functional level is set for a domain, that level of functionality applies only to that domain. Unless specified otherwise, domains are created at the *Windows 2000 mixed* functional level; forests are created at the *Windows 2000* functional level.

Table 2-1 lists the domain functional levels and the operating systems supported on the domain controllers in that domain.

Table 2-1. Domain Functional Levels

Domain functional level	Operating systems supported on the domain controllers in the domain
Windows 2000 mixed (default)	Windows NT 4 Windows 2000 Windows Server 2003
Windows 2000 native	Windows 2000 Windows Server 2003
Windows Server 2003 interim	Windows NT 4 Windows Server 2003
Windows Server 2003	Windows Server 2003

Table 2-2 lists the forest functional levels and the operating systems supported on the domain controllers in that forest:

Table 2-2. Forest Functional Levels

Forest functional level	Operating systems supported on the domain controllers in the forest
Windows 2000 (default)	Windows NT 4 Windows 2000 Windows Server 2003
Windows Server 2003 interim	Windows NT 4 Windows Server 2003
Windows Server 2003	Windows Server 2003

Before raising the forest's functional level to Windows Server 2003, verify that all domains in the forest are set to the domain functional level of Windows 2000 native or Windows Server 2003. Domains that are set to the functional level of Windows 2000 native will automatically be raised to the functional level of Windows Server 2003 at the same time the forest level is raised to Windows Server 2003.

Once the functional level, either domain or forest, has been raised, only domain controllers running those levels of operating system can be added to the domain or forest. Also, once a functional level, either domain or forest, has been raised, it cannot be reset to a lower level.

Finally, the GC is used to facilitate user logon by enabling the use of user principal names (for example, *username*@contoso.com). User principal names (UPNs) are resolved by the GC, because the GC has information about every user in every domain in the forest. The non-GC domain controllers do not possess this data, and they are unable to authenticate user logon in this format.

Operations Masters

Active Directory is designed as a multimaster replication system. This requires that all domain controllers have Write permissions for the directory database. This system works quite well for most directory operations, but for certain directory operations a single authoritative server is required. The domain controllers that perform specific roles are known as operations masters, and each has a flexible single-master operations (FSMO, pronounced *fizz-mo*) role. The five operations master roles in Active Directory are:

- Schema master
- Domain naming master
- RID master
- PDC emulator
- Infrastructure master

The first two roles, schema master and domain naming master, are *per-forest* roles. This means that there is only one schema master and only one domain naming master for every forest. The other three roles are *per-domain* roles; there is only one of these operations master roles for each domain in the forest. When you install Active Directory and create the first domain controller in the forest, it will possess all five of these roles. Similarly, as you add domains to the forest, the first domain controller in each new domain will also acquire these last three operations master roles. As you add domain controllers to a domain, you will transfer certain of these roles to other domain controllers. How you transfer these roles to other domain controllers is covered later in this chapter.

Schema Master

The schema master is the only domain controller that has Write permissions to the directory schema. To make any change to the directory schema, the administrator (who must be a member of the Schema Admins security group) must be connected to the schema master. If a modification to the schema is attempted on a domain controller other than the schema master, it will fail. After a change has been made, schema updates are replicated to all other domain controllers in the forest.

By default, the first domain controller installed in a forest (the domain controller for the forest root domain) assumes the schema master role. This role can be transferred at any time using the Active Directory Schema snap-in or by using the Ntdsutil command-line utility. The schema master is identified by the value of the *fSMORoleOwner* attribute on the schema container.

Domain Naming Master

The domain naming master is the domain controller on which new domains can be added to a forest or from which existing domains can be removed. Administrators must be connected to the domain naming master to add or remove a domain. If the domain naming master is unavailable, any attempt to add a domain to, or remove a domain from, the forest will fail.

Domains are added to a forest in one of two ways, both of which require a remote procedure call (RPC) connection to the domain naming master role holder. The most common method for creating a new domain is running Dcpromo.exe at the command line, which starts the Active Directory Installation Wizard. During this process, you have the option to install the first domain controller in a new domain. Dcpromo.exe will contact the domain naming master to make this change. When using Dcpromo.exe, if the domain naming operations master is not available, domain creation will fail. Alternatively, a new domain can be *precreated* using Ntdsutil. This utility creates a cross-reference object in the partitions container of the configuration directory partition, which is then replicated to every domain controller in the forest. Once the domain has been precreated, Dcpromo.exe can be run to create the new domain without having to contact the domain naming master.

RID Master

The RID master is a per-domain operations master role. It is used to manage the RID pool to create new security principals throughout the domain, such as users, groups, and computers. Every domain controller is issued a block of relative identifiers (RIDs) that are used to build the security identifier (SID), which uniquely identifies security principals in a domain. This block of available RIDs is called the RID pool. When the number of available RIDs in the RID pool on any domain controller in the domain begins to run low, a request is made for another block of RIDs from the RID master. It is the job of the RID master to fulfill these requests, as well as to ensure that no RID is allocated more than once. This process guarantees the unique security identity of every account in the domain.

If the RID master is unavailable for a period of time, the process of creating new accounts on specific domain controllers may be interrupted. The mechanism for requesting a new block of RIDs is designed so that this should not happen, because the request is made before all of the available RIDs in the RID pool are exhausted. However, if the RID master is offline and the requesting domain controller depletes the remainder of its RIDs, account creation will fail. To re-enable account creation, either the RID master role holder must be brought back online or the role must be transferred to another domain controller in the domain.

PDC Emulator

The PDC emulator role is required for Windows Server 2003 to coexist with pre-Windows 2000 domain controllers. In a Windows 2000 mixed functional level domain, the Windows

Server 2003 domain controller acts as the primary domain controller (PDC) for all down-level (Microsoft Windows NT versions 4 or 3.51) backup domain controllers (BDCs). In such an environment, the PDC emulator is required for processing password changes, replicating domain changes to the BDCs, and running the domain master browser service. If the PDC emulator is unavailable, all events related to these services that were initiated from down-level clients will fail.

In a Windows 2000 native or a Windows Server 2003 functional level domain, the PDC emulator still plays a role. It is used primarily to maintain password updates. All password changes made on other domain controllers in the domain are sent to the PDC emulator. If a user authentication fails on a domain controller other than the PDC emulator, authentication is retried on the PDC emulator. If the PDC emulator has accepted a recent password change for this account, authentication will succeed.

Infrastructure Master

The infrastructure master is responsible for updating the cross-domain group-to-user references. This operations master role ensures that changes made to user account names (changes to the common name attribute, *cn*) are reflected in the group membership information for groups located on a different domain. The infrastructure master maintains an up-to-date list of these references, and it then replicates this information to all other domain controllers in the domain. If the infrastructure master is unavailable, the cross-domain group-to-user references will be out of date.

Transferring Operations Master Roles

Operations master roles can be transferred either to better optimize domain controller performance or to substitute a domain controller if a role holder has become unavailable. The process for doing this will depend on the role being transferred. The following tools are used to transfer the five operations master roles:

- **Schema master** Active Directory Schema snap-in
- **Domain naming master** Active Directory Domains And Trusts administrative tool
- **RID master, PDC emulator, and infrastructure master** Active Directory Users And Computers administrative tool

To transfer an operations master role, there must be connectivity to both the current and proposed role holder domain controllers. In the event of server failure, the current role holder may not be available to effect a role transfer. In this case, the role can be seized. Seizing operations master roles is not a preferred option and should be done only if absolutely necessary. You should only seize an operations master role if it is indicated that the domain controller hosting this role will be unavailable for an extended period. More information on seizing operations master roles is provided in Chapter 15, “Disaster Recovery.”

The Schema

The schema defines every type of object that can be stored in Active Directory. Before an object can be created in Active Directory, it must first be defined in the schema. The schema also enforces a number of rules regarding the creation of objects in the database. These rules define the information that can be stored with each object and the data type of that information.

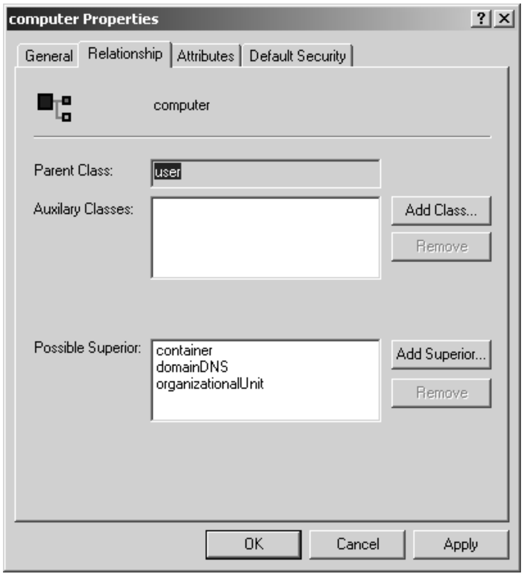
Schema Components

The schema is made up of class objects and attribute objects. The *class object* defines what new objects can be created in the directory. There must first be a class for every new object created in the directory. An example of a class object is the User class. All new user objects created in Active Directory are instances of the class "User."

The schema also defines what information can be stored for each object class. This information is defined in the schema as the *attribute object*. An object of a certain class can contain values for all of the attributes defined for that class, as well as for all of the parent classes of which this class is a child. For example, a user account can have defined attribute values for all of the objects in the User class, as well as for the User class's parent class, organizationalPerson. When creating a new user object, you can include information for that user that is defined in the schema as an attribute of all of the classes to which this new user object will belong.

Finally, the type of data that can be stored in Active Directory for each attribute is defined in the schema as the attribute's *syntax*. If the User class contains an attribute titled *displayName*, the syntax for this attribute can be defined as a string value, which accepts any alphanumeric character. The value for each attribute included with an instance of a class must meet the syntax requirements for that attribute.

The Active Directory schema supports inheritance of class objects. All schema objects are organized hierarchically in the schema naming context. Because of this hierarchical structure, any class object is able to inherit all of the characteristics of its parent class object. For example, the Computer class is actually a child of the User class. As such, the Computer class inherits all of the attributes associated with the User class. The Computer class is then associated with the attributes specific to the User class. Using the Active Directory Schema snap-in, you can see the organization of class object inheritance and the hierarchy of the object classes. Figure 2-1 illustrates the Computer class. Notice that it is a child of the User class, which is a child of the organizationalPerson class, and so on. This system of inheritance makes it much easier for administrators to create new object classes, because they do not have to define every attribute that is associated with a new class but can simply inherit all the attribute associations of a suitable parent class.



computer

Parent Class: user

Auxilary Classes:

Possible Superior: container
domainDNS
organizationalUnit

Figure 2-1. *The Computer class object in the schema, as displayed by the Active Directory Schema snap-in.*

Modifying the Schema

The Active Directory schema contains the most commonly used classes and attributes to support an enterprise directory services implementation. These attributes and classes are defined as *Category 1* objects, or *base schema* objects. In anticipation of the need to support customer-specific classes and attributes, the Active Directory schema was designed to be extensible. In other words, it can be modified, or *extended*, to include new class and attribute objects that an organization might need. Schema objects that are subsequently created are defined as *Category 2* objects. The schema is most often extended to meet the needs of an Active Directory-enabled application. A good example of this is Microsoft Exchange 2000 Server, which makes over a thousand additions to the schema to configure Active Directory to support Exchange.

Apart from using Active Directory-enabled applications, administrators can extend the schema using a variety of other methods. The schema can be extended in a batch mode using command-line administrative tools, including the LDAP Data Interchange Format Directory Exchange (LDIFDE) tool and the Comma Separated Value Directory Exchange (CSVDE) tool. The schema can also be extended programmatically, using Active Directory Service Interfaces (ADSI) and Microsoft Visual Basic scripts.

More Info For more information on either LDIFDE or CSVDE, type the command name at the command line for online help. For more information on ADSI and ADSI Edit, see the Microsoft Windows Platform software development kit (SDK), which can be downloaded or ordered on compact disc at *http://www.microsoft.com/msdownload/platformsdk/sdkupdate*. The ADSI portion of the Platform SDK can be viewed online at *http://msdn.microsoft.com/library/default.asp?url=/library/en-us/netdir/adsi /directory_services.asp*.

Finally, the schema can be modified from the Windows Server 2003 user interface (UI) using the Active Directory Schema snap-in. To use the Active Directory Schema snap-in, you must first register the snap-in by executing the *Regsvr32 Schmmgmt.dll* command from the command line. You must be a member of the Schema Admins global group to modify the schema using this interface.

To understand how modifying the schema works, imagine that an organization needed to keep records of employee start dates. It would have to maintain the employee start date as an attribute of the user object in Active Directory. To have this attribute available when each new user object is to be created, the attribute would first be defined in the schema.

➲ **To use the Active Directory Schema snap-in to add a new attribute to the schema and associate it with the User class object, perform the following steps:**

1. Open the Active Directory Schema snap-in.
2. Select the Attributes folder in the tree pane.
3. From the Action menu, click Create Attribute.
4. At the Schema Object Creation warning dialog, click Continue.
5. In the Create New Attribute dialog box, supply information for the Identification section:
    - Common Name
    - LDAP Display Name
    - Unique X500 Object ID
    - Description
6. In the Syntax And Range section, supply information for:
    - Syntax
    - Minimum
    - Maximum
7. Select whether or not the new attribute is a Multi-Valued attribute.

Further information regarding the content of each field is available by selecting the field's text box, then pressing the F1 function key.

Obtaining an X500 Object ID

One of the concerns about modifying the schema is the possibility that two applications will make incompatible modifications to the schema. Each object in the schema must be unique. To manage this concern, every class and attribute in Active Directory can be identified by a unique object identifier (OID). The goals of the OID are to be able to uniquely identify any object or attribute in Active Directory and to ensure that no other schema object uses the same OID.

To accomplish this identification, organizations planning to create new OIDs should register with the International Standards Organization (ISO) or the American National Standards Institute (ANSI). When you register, the standards organization assigns you part of the OID space, which you can then extend to suit your needs. For example, your company may be granted a number such as 1.2.840.*xxxx*. This number is arranged hierarchically and can be broken down into:

1–ISO

    2–ANSI

        840–United States

            xxxx–A unique number identifying your company

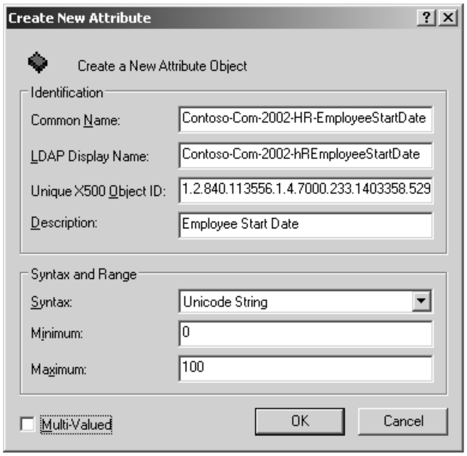
Once you have been granted the number, you can manage your own part of the hierarchy. For example, if you create a new attribute called *Employee Start Date*, you could assign it a number like 1.2.840.*xxxx*.12.

To understand how this works, imagine for example that the OID for a contact in Active Directory is 1.2.840.113556.1.5.15. The first three parts of the number have been assigned to ISO, ANSI, and the United States, respectively. ANSI then assigned 113556 to Microsoft, who assigned 1 to Active Directory, 5 to Active Directory classes, and 15 to the Contact class.

The *Microsoft Windows 2000 Server Resource Kit*, available from Microsoft Press, includes a tool called OIDGen that can be used to create a unique OID for object classes or attributes without registering the OIDs. This tool should not be used if you are making changes to a schema that will ever be deployed outside of your organization. For external deployment, Microsoft offers to generate and register your new OIDs. See *http://msdn.microsoft.com/certification/ad-registration.asp* for further details.

Figure 2-2 shows the creation of a new attribute using the Active Directory Schema snap-in.

Note Adding a new attribute to the schema does not mean that the attribute will automatically be accessible from any of the administrative tools. The administrative tools like Active Directory Users And Computers only show some of the attributes for each class and do not show any attributes you add. If you want the new attribute to appear in an administrative tool, you must either modify the existing tool or create your own. For information on how to modify and create administrative tools, see the Directory Services section of the Platform SDK at *http://msdn.microsoft.com/library/default.asp?url=/library/en-us/netdir/ad/extending_the_user_interface_for_directory_objects.asp.*



Common Name: Contoso-Com-2002-HR-EmployeeStartDate
LDAP Display Name: Contoso-Com-2002-hREmployeeStartDate
Unique X500 Object ID: 1.2.840.113556.1.4.7000.233.1403358.529
Description: Employee Start Date
Syntax: Unicode String
Minimum: 0
Maximum: 100
☐ Multi-Valued

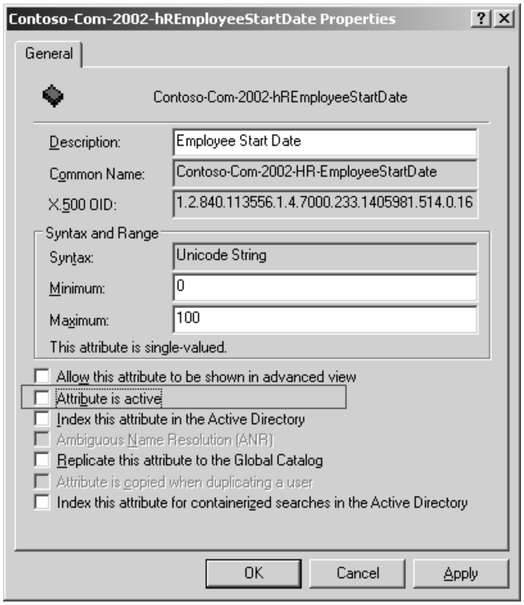
Figure 2-2. *Creating a new schema attribute.*

Deactivating Schema Objects

While extending the schema is a straightforward operation, careful planning should be done before implementing such changes. Once the schema has been extended, or an existing class or attribute has been modified, these changes are not reversible. Objects in the schema cannot be deleted. If you do make an error when extending the schema, you may choose to disable (deactivate) the object. In Windows Server 2003, schema objects that are deactivated can be used again if necessary, and new schema objects can be created with the same name as a deactivated object.

There are several points to keep in mind regarding deactivating schema class and attribute objects. First, you can only deactivate classes and attributes that you have specifically created—that is, *Category 2* objects. You cannot deactivate a *Category 1*, or *base schema*, object. Second, you cannot deactivate an attribute that is a member of a class that is not also deactivated. This restriction prevents errors in creating new instances of the non-deactivated class if the deactivated attribute is a required attribute.

To deactivate either a *Category 2* class or an attribute object, set the Boolean value of the *isDefunct* attribute of the schema object to *true*. This can be accomplished by using a tool such as ADSI Edit or by using the Active Directory Schema snap-in. Figure 2-3 illustrates the setting that must be unchecked to deactivate the *EmployeeStartDate* attribute created in the example given earlier.



Contoso-Com-2002-hREmployeeStartDate Properties

Contoso-Com-2002-hREmployeeStartDate

Description: Employee Start Date
Common Name: Contoso-Com-2002-HR-EmployeeStartDate
X.500 OID: 1.2.840.113556.1.4.7000.233.1405981.514.0.16

Syntax and Range
Syntax: Unicode String
Minimum: 0
Maximum: 100
This attribute is single-valued.

☐ Allow this attribute to be shown in advanced view
☐ Attribute is active
☐ Index this attribute in the Active Directory
☐ Ambiguous Name Resolution (ANR)
☐ Replicate this attribute to the Global Catalog
☐ Attribute is copied when duplicating a user
☐ Index this attribute for containerized searches in the Active Directory

Figure 2-3. *Using the Active Directory Schema snap-in to deactivate a schema attribute.*

After a schema object has been deactivated, it is treated in all respects as if it does not exist. The error messages that are returned if an attempt is made to create a new instance of a defunct class or attribute are the same as when there is no existing class or attribute in the schema. Additionally, the only modification that can be made to a deactivated schema object is to reactivate it. To reactivate the defunct schema object, simply set the *isDefunct* attribute to *false*. After a defunct schema object has been reactivated, it can be used again to create new instances of the class or attribute. There are no adverse effects of this deactivation/reactivation process.

Active Directory Logical Structure

After you install Active Directory in your network environment and begin to implement the appropriate Active Directory design for your business purposes, you will be working with the logical structure of Active Directory. (This logical structure is the directory services model that defines every security principal in the enterprise as well as the organization of these security principals.) The Active Directory database contains the following structural objects:

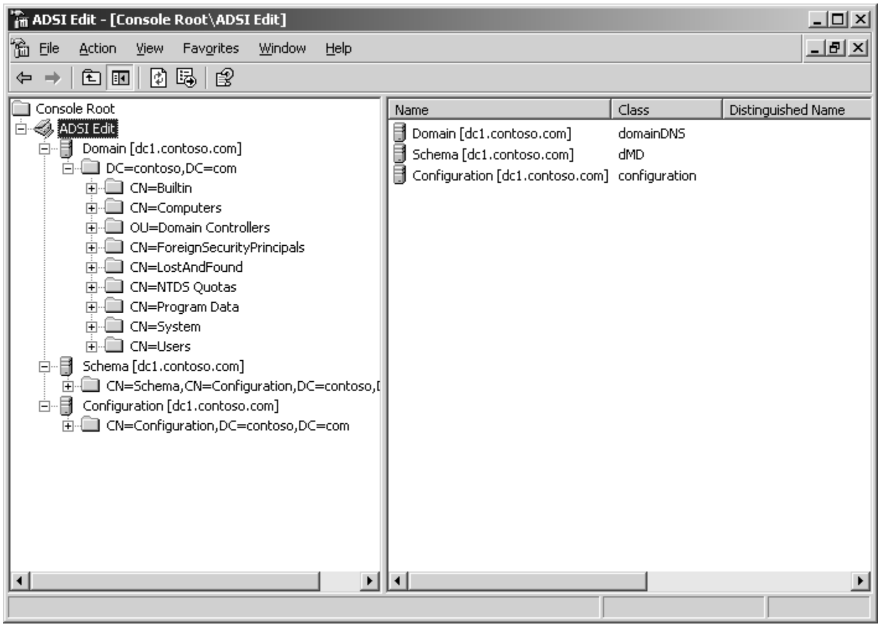
- Partitions
- Domains
- Domain trees

- Forests
- Sites
- Organizational units

This section provides an introduction to these components. It will also discuss the concept of trusts, which are used to enable access to resources for security principals that are stored in different domains. In Chapter 5 you will learn how and why these structural components are used to achieve specific business goals (such as secured access to resources) and optimize network performance. Security principals themselves, such as users, groups, and computers, are not discussed in this chapter.

Active Directory Partitions

As described earlier, the Active Directory database is stored in one database file on the hard disk of each domain controller. The directory database is divided into multiple logical partitions, with each partition storing different types of information. Active Directory partitions are also called *naming contexts* (NCs). Active Directory partitions are visible through use of a tool such as Ldp.exe or ADSI Edit, as shown in Figure 2-4.



Name	Class	Distinguished Name
Domain [dc1.contoso.com]	domainDNS	
Schema [dc1.contoso.com]	dMD	
Configuration [dc1.contoso.com]	configuration	

Figure 2-4. *Active Directory partitions that are visible using the ADSI Edit tool.*

Domain Directory Partition

The domain directory partition is the partition where most of the action takes place. This partition contains all of the domain information, including information about users, groups, computers, and contacts. Essentially, anything that can be viewed through the Active Directory Users And Computers administrative tool is stored in the domain directory partition.

The domain directory partition is automatically replicated to all domain controllers in the domain. The partition contains the information that each domain controller needs to authenticate users.

Configuration Directory Partition

The configuration directory partition contains the information about the configuration of the entire forest. For example, all of the information about sites, site links, and replication connections are stored in the configuration directory partition. Many application programs also store information in the configuration partition. Exchange 2000 Server stores all of its configuration information in the Active Directory configuration directory partition rather than in its own directory service, as did previous versions of Exchange. Other applications, like Microsoft Internet Security And Acceleration (ISA) Server, can also store configuration information in the configuration directory partition. When you install the first ISA server in your organization, you can configure an array that will store all of the ISA configuration information in Active Directory. Additional ISA servers can then easily be installed to use exactly the same configuration. The server configuration is read from Active Directory.

Because the configuration directory partition contains information about the entire forest, it is also replicated throughout the entire forest. Each domain controller contains a writable copy of the configuration directory partition, and changes to this directory partition can be made on any domain controller in the organization. This means that the configuration information is then replicated to all the other domain controllers. When the replication is fully synchronized, every domain controller in the forest will have the same configuration information.

Schema Directory Partition

The schema directory partition contains the schema for the entire forest. As described earlier in this chapter, the schema is a set of rules detailing what types of objects can be created in Active Directory as well as rules about each type of object.

The schema directory partition is replicated to all domain controllers in the entire forest. However, only one domain controller, the schema master, has a writable copy of the schema directory partition. All changes to the schema must be made on the schema master; the changes are then replicated to all other domain controllers.

Global Catalog Partition

The GC partition is not a partition in the same sense as the other partitions. The GC partition is stored in the database like the other partitions, but administrators cannot enter information directly into this partition. The GC is a read-only partition on all GC servers, and it is built from the contents of the domain databases. Each attribute in the schema has a Boolean value named *isMemberOfPartialAttributeSet*. If this value is set to *true*, the attribute is replicated to the GC.

Application Directory Partitions

The last type of partition in Windows Server 2003 Active Directory is the application directory partition. Only one type of application directory partition is created by default in Active Directory—for the Domain Name System (DNS) server service. Installing the first Active Directory integrated zone creates the ForestDnsZones and the DomainDnsZones application directory partitions. Application directory partitions can store any type of Active Directory object except security principals. Also, because application directory partitions are created to control where the data is replicated, none of the objects in the application directory partition can be replicated to the GC partition.

Application directory partitions are used to store application-specific information. The advantage of application directory partitions is that replication of the information in the partition can be controlled. Often the information might be fairly dynamic, so you want to control the replicas on the network to limit the amount of replication traffic that is created on the network. When the application directory partition is created, you can configure which domain controllers will receive a replica of the partition. The domain controllers that receive a replica of the application directory partition can be in any domain or site in the forest.

The naming scheme for application directory partitions is identical to other Active Directory directory partitions. For example, the DNS name for the configuration directory partition in the Contoso.com forest is dc=Configuration, dc=Contoso, dc=com. If you create an application directory partition called AppPartition1 in the Contoso.com domain, its DNS name is dc=AppPartition1, dc=Contoso, dc=com. Application directory partitions are quite flexible in regard to where you can create the partition, or more accurately, what the naming context for the partition will be. For example, you can create an additional application directory partition in the AppPartition1 partition resulting in a partition with a name of dc=AppPartition2, dc=AppPartition1, dc=Contoso, dc=com. You can even create an application directory partition with a DNS name that is not contiguous with any domain in the forest. You can create an application directory partition in the Contoso.com domain that has a DNS name of dc=AppPartition. In effect, this creates a new tree in the forest.

Note Choosing the DNS name for the application namespace does not affect the functionality of the application directory partition in any way. The only difference will be in the configuration of the LDAP client that is accessing the partition. Application directory partitions are designed for LDAP access, so the client must be configured to search the right namespace on the server.

One of the complicating factors when creating an application directory partition is maintaining permissions to the objects in the partition. With the default partitions in Active Directory, the permissions are automatically assigned. When an object is created in the domain directory partition, the Domain Admins group is automatically assigned full permissions to the object. When an object is created in the configuration directory partition or schema directory partition, user and group accounts from the forest root domain are assigned permissions. Because an application directory partition can be created in any domain directory partition or even as a separate tree in the forest, this default way of assigning permissions does not apply. While it is easy to assign a group like Domain Admins full control of the objects in the partition, what is not clear is which domain is the default domain. To deal with this issue, application directory partitions are always created with a security descriptor reference domain. This domain becomes the default domain that is used to assign permissions to objects in the application directory partition. If an application directory partition is created in a domain directory partition, the parent domain is used as the security descriptor reference domain, in effect creating an inheritance of permissions. If the application directory partition creates a new tree in the forest, the forest root domain is used as the reference domain.

Tip Normally, the application directory partitions will be created by the installation of an application that requires the use of an application directory partition. Also, the application installation procedure should allow for the creation of additional replicas on other domain controllers. While you can create application directory partitions using Ntdsutil, you would normally not expect to do this in a production environment. The procedures for managing application directory partitions are described in the Windows Server 2003 Help And Support Center. For detailed information on application directory partitions, including how to access them programmatically, search for "Using application directory partitions" on *msdn.microsoft.com*.

Once an application directory partition has been created with multiple replicas, the replication management of the partition is handled in exactly the same way that replication is handled for all other partitions. For more information on Active Directory replication, see Chapter 4.

Domains

The domain is the most basic building block in the Active Directory model. When you install Active Directory on your first computer running Windows Server 2003, you create a domain. A domain serves as an administrative boundary, and it also defines the boundary of certain security policies. Each domain has at least one domain controller; optimally it will have two or more.

Active Directory domains can be hierarchically organized. The first domain in the enterprise is known as the *forest root domain*—commonly referred to as either the *root domain* or the *forest domain*—and it is the starting point for an Active Directory namespace. For example, the first domain in the Contoso organization is Contoso.com. This first domain can either be a *dedicated* or a *non-dedicated* root domain. A dedicated root, also known as an *empty root*, is one that is used as an empty placeholder to start Active Directory. This domain will contain no live user or group accounts, and it will not be used to assign access to resources. The only accounts that are contained in the dedicated root domain are the default domain user and group accounts, such as the Administrator account and the Domain Admins global group. A non-dedicated root domain is one in which actual user and group accounts are created. The reasons for selecting either a dedicated or non-dedicated forest root are discussed in Chapter 5.

All other domains in the enterprise exist either as *peers* to the root domain or as *child* domains. Peer domains exist at the same hierarchical level as the root domain. Figure 2-5 illustrates the peer domain model.

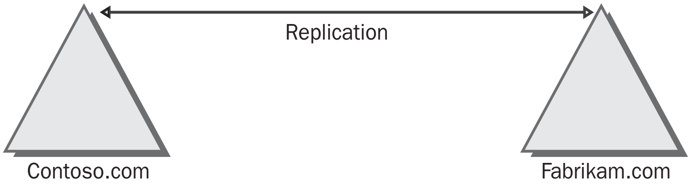


Figure 2-5. *Active Directory domains organized as peers.*

Alternatively, and more commonly, domains installed subsequent to the root domain are installed as child domains. *Child domains* share the same Active Directory namespace as the parent domain (the root domain). For example, if the first domain in the Contoso organization is named Contoso.com, a child domain in this structure might be named NAmerica.Contoso.com. The NAmerica.Contoso.com domain would be created to manage all of the security principals for the North American locations of the Contoso organization. If the organization is sufficiently large or complex, additional child domains, such as Sales.NAmerica.Contoso.com, might be required. Figure 2-6 illustrates the parent-child domain hierarchy for the Contoso organization.

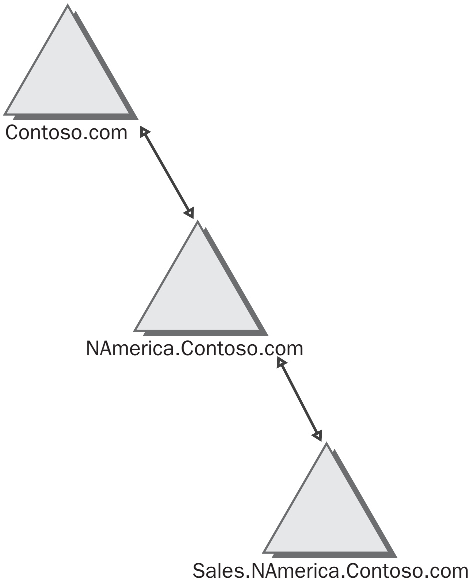


Figure 2-6. *Parent-child domain model for Contoso Corp.*

Domain Trees

As subsequent domains are created in the Active Directory infrastructure, they can either share the existing Active Directory namespace or they can have a separate namespace. To create a separate namespace for the new domain, a new domain tree is created. Regardless of whether a single namespace or multiple namespaces are used, additional domains in the same forest function in exactly the same way. The creation of additional domain trees is purely an organizational and naming decision, not one that affects functionality. A domain tree contains at least one domain. Even a single-domain organization has a domain tree. Using multiple trees rather than child domains does have an impact on the DNS configuration (as discussed in Chapter 3, "Active Directory and Domain Name System").

A domain tree results when an organization creates a domain subsequent to the forest root domain but does not want to use an existing namespace. In the Contoso example, if the existing domain tree is using the namespace of Contoso.com, a new domain can be created that uses a completely different namespace, such as Fabrikam.com. If further domains are required to satisfy the Fabrikam business unit, they can be created as children of the Fabrikam domain tree. See Figure 2-7 for an illustration of the Contoso organization with multiple domain trees.

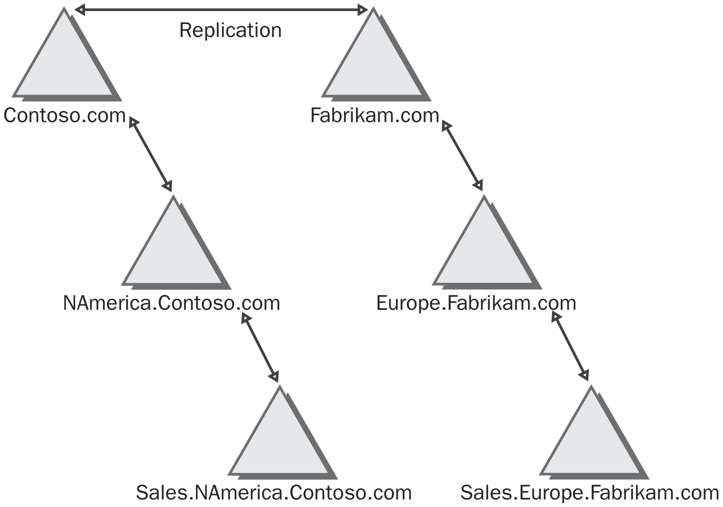


Figure 2-7. *Contoso Corp. with multiple domain trees.*

Forests

The *forest* is the ultimate security boundary for the enterprise. All domains and domain trees exist within one or more Active Directory forests. A forest is the outermost replication and security boundary for the organization.

An Active Directory forest can be defined by what is shared by all domain controllers in the forest. The shared components include:

- **A common schema** All domain controllers in the forest will have the same schema. The only way to deploy two different schemas in your organization is to deploy two separate forests.
- **A common configuration directory partition** All domain controllers in the forest have the same configuration container, which is useful for Active Directory operations like replication within the forest. The configuration directory partition is also used extensively by Active Directory-enabled applications like Exchange 2000 Server and ISA.
- **A common GC** The GC contains information about all of the objects in the entire forest. This makes searching for any object in the forest efficient and enables users to log on in any domain in the forest using their UPN.

- **A common set of forest-wide administrators** Two security groups are created in the root domain for the forest, and these groups have permissions not granted to any other users. The Schema Admins group is the only group that has the right to modify the schema, and the Enterprise Admins group is the only group that has the right to perform forest-level actions such as adding or removing domains from the forest. The Enterprise Admins group is also automatically added to each local Administrators group on the domain controllers in every domain in the forest.
- **A shared trust configuration** All the domains in the forest are automatically configured to trust all the other domains in the forest. There is more on trusts in the next section.

Figure 2-8 shows the Contoso forest.

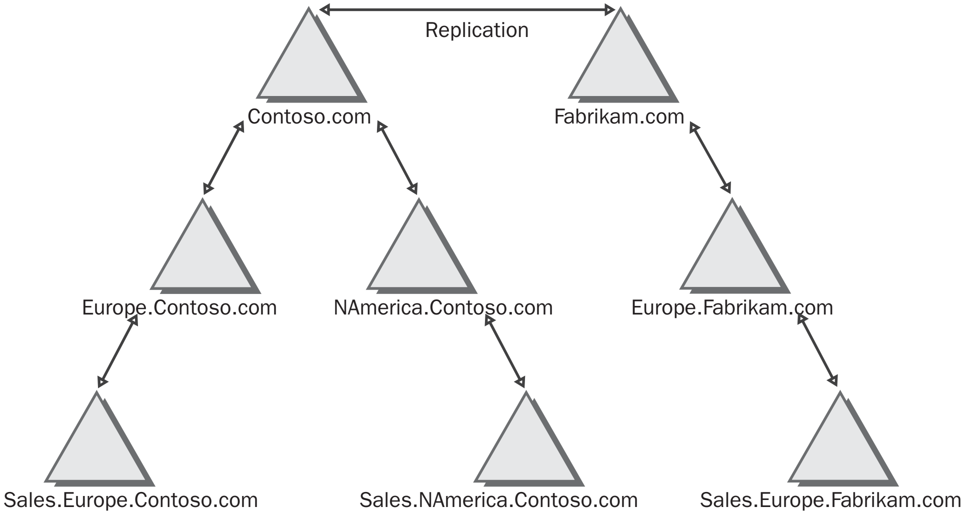


Figure 2-8. *The Contoso forest.*

Trusts

By default, the domain is the boundary of resource access in an organization. With sufficient permissions, any security principal (for example, a user or group account) can access any shared resource in the same domain. In order for security principals to access shared resources that exist outside of their domain, Active Directory trust relationships are utilized. A *trust* is an authentication connection between two domains by which security principals can be authorized to access resources on the other domain. There are several types of trust relationships, including:

- Transitive trusts
- One-way trusts
- Forest trusts
- Realm trusts

Transitive Trusts

All domains in a tree maintain transitive, two-way trust relationships with every other domain in that tree. In the example provided earlier, when the NAmerica.Contoso.com domain is created as a child domain of the root domain Contoso.com, an automatic two-way trust is created between the NAmerica.Contoso.com and the Contoso.com domains. Through this trust, any user in the NAmerica.Contoso.com domain can access any resource in the Contoso.com domain to which permission has been granted. Likewise, if any security principals exist in the Contoso.com domain (as in a non-dedicated root domain), they can be given access to resources in the NAmerica.Contoso.com domain.

Within a forest, the trusts are set up as either parent-child trusts or as tree root trusts. An example of a parent-child trust is the trust between the NAmerica.Contoso.com domain and the Contoso.com domain. A *tree root trust* is the trust between two trees in the forest, for example, between Contoso.com and Fabrikam.com.

However, all of the trusts between domains in a forest are also *transitive*. The transitive nature of the trust means that all the domains in the forest trust each other. If the Contoso.com domain trusts the NAmerica.Contoso.com domain, and the Europe.Contoso.com domain trusts the Contoso.com domain, then transitivity indicates that the Europe.Contoso.com domain also trusts the NAmerica.Contoso.com domain. Therefore, users in the NAmerica.Contoso.com domain can access resources in the Europe.Contoso.com domain and vice versa. The transitive trusts also apply to the tree root trusts. The NAmerica.Contoso.com domain trusts the Contoso.com domain, and the Contoso.com domain trusts the Fabrikam.com domain. Therefore, the NAmerica.Contoso.com domain and the Fabrikam.com domain also share a transitive-trust relationship.

One-Way Trusts

In addition to the automatic, two-way transitive trusts that are created when a new child domain is created, one-way trusts can be created between domains in the forest. One-way trusts might be created to enable resource access between domains that are not in a direct trust relationship. One-way trusts might also be used to optimize performance for domains that are connected through transitive trusts. These one-way trusts are called *shortcut trusts*. A shortcut trust is desirable when there is frequent resource access between domains that are remotely connected through the domain tree or forest. As an example, imagine the Contoso forest as illustrated as Figure 2-9.

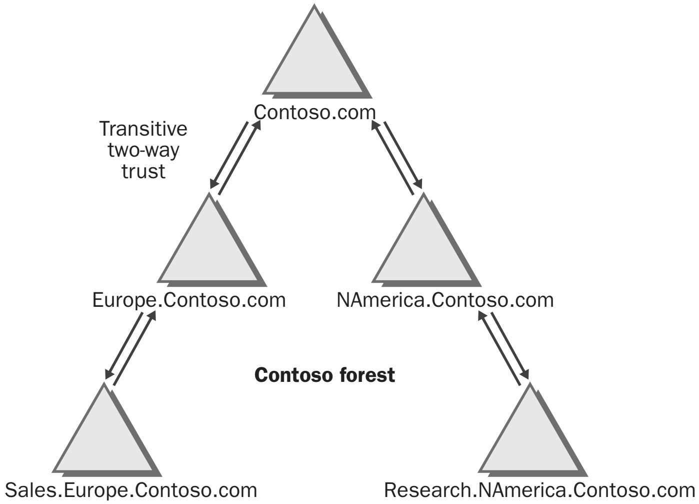


Figure 2-9. *Trusts in the Contoso forest.*

If a security group in the Sales.Europe.Contoso.com domain has a frequent need to access a shared resource in the Research.NAmerica.Contoso.com domain, and with only transitive trusts established between the domains, users in the Sales.Europe.Contoso.com domain must authenticate through every domain in the tree between them and the domain that contains the resource. This is not efficient if the need is frequent. A shortcut trust is a direct, one-way trust that will efficiently enable users in the Sales.Europe.Contoso.com domain to authenticate in the Research.NAmerica.Contoso.com domain—without traversing the entire directory tree to get there. Figure 2-10 illustrates this shortcut trust. If it is determined that there is need to provide this same trust, but in the other direction, another shortcut trust can be created between the two domains, with their two roles reversed. (Such a double shortcut trust has the appearance of a transitive trust, but this unique trust relationship does not extend beyond these two domains.)

Forest Trusts

Forest trusts are a new feature in Windows Server 2003. A *forest trust* is a two-way transitive trust between two separate forests. With a forest trust, security principals in one forest can be given access to resources in any domain in a completely different forest. Also, users can log on to any domain in either forest using the same UPN.

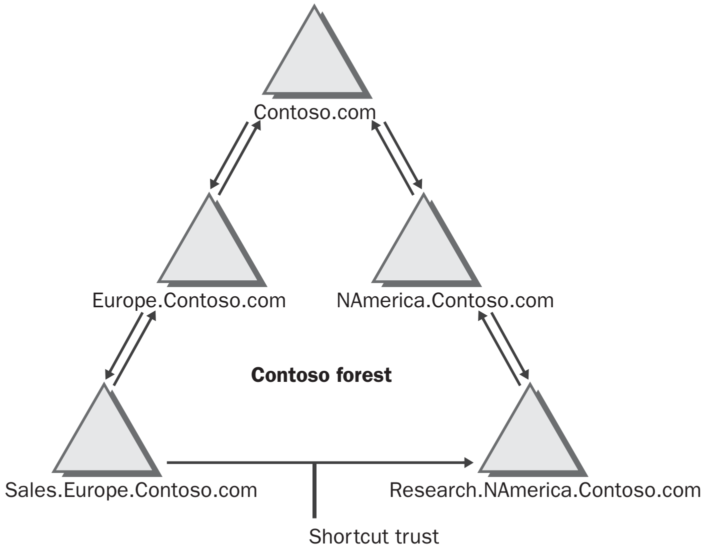


Figure 2-10. *A shortcut trust from the Sales.Europe.Contoso.com domain to the Research.NAmerica.Contoso.com domain.*

Forest trusts can be very useful in a Windows Server 2003 environment. If an organization requires more than one forest for political or technical reasons, the use of a forest trust means that it is easy to assign access to resources across all the domains, regardless of which forest the user or resource is in. If two companies that have deployed Windows Server 2003 forests merge, the two forests can be logically joined by using the trust.

While forest trusts do provide some excellent functionality, they are also subject to some limitations:

- Forest trusts are not transitive to other forests. For example, if Forest1 has a forest trust with Forest2, and Forest2 has a forest trust with Forest3, Forest1 does not automatically have a forest trust with Forest3.
- Forest trusts only make authentication possible between forests; they do not provide any other functionality. For example, each forest will still have a unique GC, schema, and configuration directory partition. No information is replicated between the two forests—the forest trust just makes it possible to assign access to resources between forests.
- In some cases, you may not want to have all the domains in one forest trust all the domains in another forest. If this is the case, you can set up one-way, non-transitive trusts between individual domains in two separate forests.

Figure 2-11 illustrates a forest trust in the Contoso enterprise.

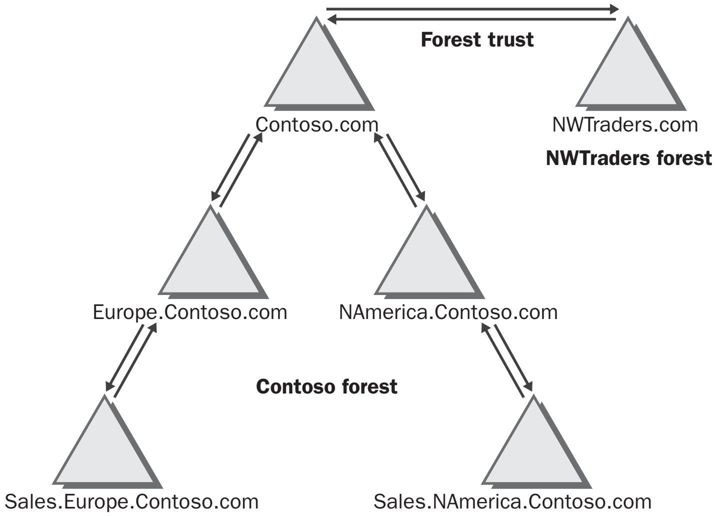


Figure 2-11. *A forest trust in the Contoso enterprise connects the Contoso.com and NWTraders.com domains, each residing in a different forest.*

Realm Trusts

The last type of trust is a *realm trust*. A realm trust is configured between a Windows Server 2003 domain or forest and a non-Windows implementation of a Kerberos v5 realm. Kerberos security is based on an open standard, and there are several other implementations of Kerberos-based network security systems available. Realm trusts can be created between any Kerberos realms that support the Kerberos v5 standard. Realm trusts can be either one-way or two-way, and they can also be configured to be transitive or non-transitive.

Sites

All of the Active Directory logical components discussed so far are almost completely independent of the physical infrastructure for your network. For example, when you design the domain structure for a corporation, where the users are located is not the most important question you need to ask. All the users in a domain may be located in a single office building, or they may be located in offices around the world. This independence of the logical components from the network infrastructure comes about largely as a result of the use of sites in Active Directory.

Sites provide the connection between the logical Active Directory components and the physical network infrastructure. A *site* is defined as an area of the network where all domain controllers are connected by a fast, inexpensive, and reliable network connection. In most cases, a site contains one or more Internet Protocol (IP) subnets connected on a local area network (LAN) or very high-speed wide area network (WAN) and connected to the rest of the network with slower WAN connections.

The primary reason for creating sites is to be able to manage any network traffic that must use slow network connections. Sites are used to control network traffic within the Windows Server 2003 network in three different ways:

- **Replication** One of the most important ways that sites are used to optimize network traffic is in the management of replication traffic between domain controllers and GC servers. For example, within a site, any change made to the directory will be replicated within about 5 minutes. The replication schedule between sites can be managed so that the replication traffic will occur during non-working hours. By default, replication traffic between sites is compressed to conserve bandwidth, while replication traffic within a site is not compressed. (Chapter 4 goes into much more detail on the differences between intersite and intrasite replication.)
- **Authentication** When a user logs on to a Windows Server 2003 domain from a Windows 2000 or Microsoft Windows XP Professional client, the client computer will always try to connect a domain controller in the same site as the client. As discussed in Chapter 3, every domain controller registers site-specific service locator (SRV) records—when the client computer tries to locate a domain controller, it will always query the DNS servers for these site records. This means that the client logon traffic will remain within the site. If the domain is operating at the Windows 2000 native functional level or the Windows Server 2003 functional level, the client will also try to locate a GC during logon. If there is a GC server in the site, the client will also connect to that server. (The role of sites in locating domain controllers is discussed in detail in Chapter 3.)

Note Client computers running Windows NT 4 SP6a can log onto Active Directory domain controllers if they have installed the Directory Services Client, which is available for download at *http://www.microsoft.com/windows2000/server/evaluation/news/bulletins/adextension.asp*. For those clients that have not been upgraded from Windows 95 or Windows 98, the Directory Services Client software is available on the Windows 2000 Server compact disc.

- **Site-aware network services** The third way that sites can preserve network bandwidth is by limiting client connections to site-aware applications and services on the site. For example, by using Distributed File System (DFS), you can create multiple replicas of a folder in different sites on the network. Because DFS is designed to be aware of the site configuration, client computers always try to access a DFS replica in their own site before crossing a WAN link to access the information in another site.

Every computer on a Windows Server 2003 network will be assigned to a site. When Active Directory is installed in a Windows Server 2003 environment, a default site called Default-

First-Site-Name is created, and all computers in the forest will be assigned to that site unless additional sites are created. When additional sites are created, the sites are linked to IP subnets. When a server running Windows Server 2003 is promoted to become a domain controller, the domain controller is automatically assigned to a site that corresponds to the computer's IP address. If needed, domain controllers can also be moved between sites using the Active Directory Sites And Services administrative tool.

Client computers determine their sites the first time they start up and log on to the domain. Because the client computer does not know which site it belongs to, it will connect to any domain controller in the domain. As part of this initial logon process, the domain controller will inform the client which site it belongs to, and the client will cache that information for the next logon.

Note If a domain controller or a client computer has an IP address that is not linked to a specific site, that computer will be placed in the Default-First-Site-Name site. Every computer that is part of a Windows Server 2003 domain must belong to a site.

As mentioned earlier in this chapter, there is no direct connection between sites and the other logical concepts in Active Directory. One site can contain more than one domain, and one domain can cross multiple sites. For example, as shown in Figure 2-12, the Seattle site contains both the Contoso.com domain and the NAmerica.Contoso.com domain. The NWTraders.com domain is spread across multiple sites.

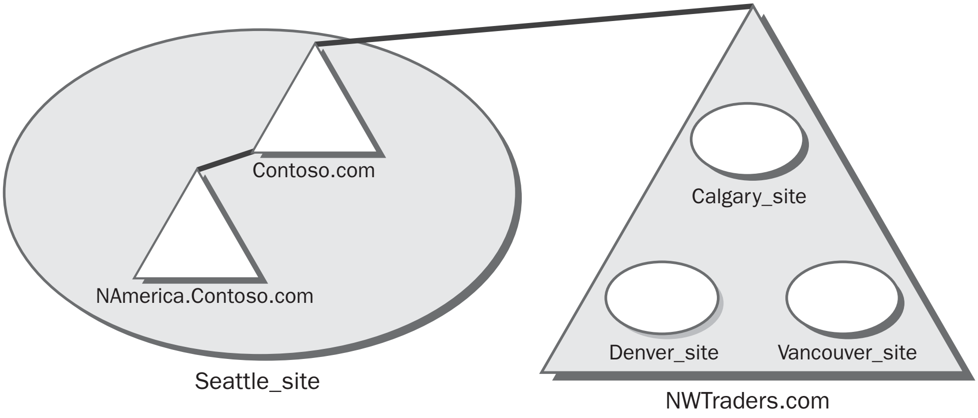


Figure 2-12. *Sites and domains within an Active Directory forest.*

Note Sites are discussed in more detail in several other chapters in this book. Chapter 3 details the role of DNS and sites for client logons. Chapter 4 addresses the role of sites in replication and how to create and configure sites. Chapter 5 goes into detail on designing an optimal site configuration for an Active Directory forest.

Organizational Units

By implementing multiple domains in a forest, either in a single tree or in multiple trees, Windows Server 2003 Active Directory can scale to provide directory services for almost any size network. Many of the components of Active Directory, such as the global catalog and automatic transitive trusts, are designed to make the use and management of this enterprise directory efficient regardless of how big the directory gets.

Organizational units (OUs), however, are designed to make Active Directory easier to administer. Rather than dealing with how to manage multiple Active Directory domains, OUs are used to make the management of single domains more efficient. OUs are used to create a hierarchical structure within a domain. A domain might contain thousands of objects (or even hundreds of thousands). Managing this many objects without some means of organizing the objects into logical groupings is very difficult. OUs provide exactly this functionality. Figure 2-13 shows an example of what the OU structure might look like at Contoso.

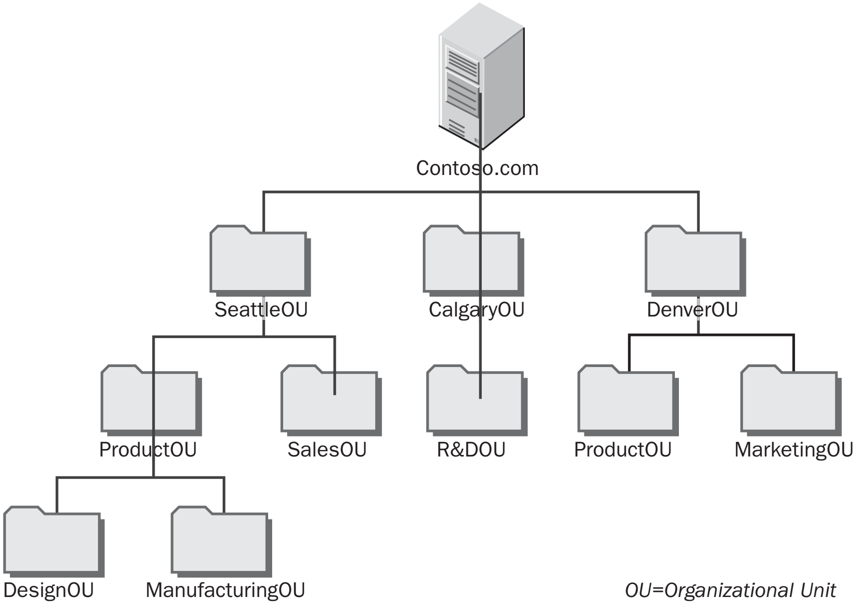


Figure 2-13. *A sample OU structure*

OUs are container objects that can contain several types of directory service objects, including:

- Computers
- Contacts
- Groups

- inetOrgPerson
- Printers
- Users
- Shared folders
- Organizational units

OUs are used to group objects together for administrative purposes. There are two ways that OUs can be used as administrative units: to delegate administrative rights and to manage a group of objects as a single unit.

Using OUs to Delegate Administrative Rights

OUs can be used to delegate administrative rights. For example, a user can be given the rights to perform administrative tasks for a specific OU. These rights could be high-level rights where the user has full control of the OU and can do anything in that OU, or the rights can be very limited and specific (such as only being able to reset passwords for users in that OU). The user that has been given administrative rights to an OU does not by default have any administrative rights outside the OU.

The OU structure is very flexible for assigning rights (also called permissions in many Windows dialog boxes and Properties sheets, and in other portions of this book) to objects inside the OU. The OU itself has an access control list (ACL) where you can assign rights for that OU. Each object in an OU and, in fact, each attribute for each object, also has an ACL. This means that you can have extremely precise control of the administrative rights anyone can have in the OU. For example, you can give a Help Desk group the right to change passwords for users in an OU but not to change any other properties for the user accounts. Or you can give the Human Resources department the right to modify any personal information on all user accounts in all OUs, but not give them any other rights to any other objects.

Using OUs to Administer Groups of Objects

Another reason for using OUs is to group objects together so that the objects can all be administered the same way. For example, if you want to administer all of the workstations in a department the same way (such as limiting which users have the right to log on to the workstations) you can group all the workstations into an OU and configure the Logon Locally permission at the OU level. This permission is applied to all workstations in that OU. As another example of grouping objects for administrative purposes, if a collection of users needs the same standard desktop configuration and the same set of applications, the users can be put into an OU and group policies can then be used to configure the desktop and to manage the installation of applications.

In many cases, objects in an OU will be managed through group policies. The Group Policy Object Editor is a management tool that can be used to manage each user's working environment. Group policies can be used to lock down user desktops, to give them a standardized desktop, to provide logon and logoff scripts, and to provide folder redirection. Table 2-3 provides a brief list of the types of settings available in the Group Policy Object Editor.

Table 2-3. Group Policy Setting Types

Setting types	Explanation
Administrative templates	Used to manage registry-based parameters for configuring application settings and user desktop settings, including access to the operating system components, access to control panel, and configuration of offline files.
Security	Used to manage the local computer, domain, and network security settings, including controlling user access to the network, configuring account policies, and controlling user rights.
Software installation	Used to centralize the management of software installations and maintenance.
Scripts	Used to specify scripts that can be run when a computer starts or shuts down, or when a user logs on or off.
Folder redirection	Used to store certain user profile folders on a network server. These folders, such as the My Documents folder, appear to be stored locally but are actually stored on a server where they can be accessed from any computer on the network.

Group policies will be most commonly assigned at the OU level. This eases the task of administering the users in the OU because you can assign one Group Policy Object (GPO)—for example, a software installation policy—to the OU, which is then enforced on all the users or computers in the OU.

Caution OUs are not security principals. This means that you cannot use an OU to assign permissions to a resource and then have all of the users in the OU automatically inherit those permissions. OUs are used for administrative purposes and to grant access to resources, so you will still need to use groups.

Summary

This chapter introduced the basic physical and logical components of Active Directory in Windows Server 2003. While having an understanding of the physical components is important (especially when dealing with database management, domain controller placement, and schema management), most of the work you will do in Active Directory will be with the logical components. Most of the rest of this book deals with the logical structure of Active Directory.

Chapter 3

Active Directory and Domain Name System

Microsoft Windows Server 2003 Active Directory directory service relies entirely on Domain Name System (DNS) to locate resources on a network. Without a reliable DNS infrastructure, domain controllers on your network will not be able to replicate with each other, your Microsoft Windows 2000 and Microsoft Windows XP Professional clients will not be able to log on to the network, and your servers running Microsoft Exchange 2000 Server will not be able to send e-mail. Essentially, if your DNS implementation is not stable, your Windows Server 2003 network will fail. This means you must have a thorough knowledge of DNS concepts and the Windows Server 2003 implementation of DNS if you are going to manage a Windows Server 2003 Active Directory environment.

This chapter begins by providing a brief overview of DNS as a service. This section is not specific to Windows Server 2003 environments, but is necessary for understanding DNS. The chapter then goes into detail about why Active Directory depends so heavily on DNS and how the name resolution process works. The final section of the chapter focuses on the DNS service in Windows Server 2003, Standard Edition; Windows Server 2003, Enterprise Edition; and Windows Server 2003, Datacenter Edition. The Windows Server 2003 operating system brings several excellent features to DNS that can greatly enhance the deployment of Active Directory.

Note Windows Server 2003, Web Edition, does not require, nor does it support, Active Directory.

DNS Overview

DNS is a name resolution service. If you are trying to find a server on the Internet, you are much more likely to remember a name like www.microsoft.com than an Internet Protocol (IP) address like 207.46.230.219. However, in order for your computer to connect to the Microsoft Web site, it needs to know the IP address. DNS performs that translation—you provide your browser with the name of the computer that you would like to connect to, and DNS resolves that name to the correct IP address.

Note Because DNS is essential for Active Directory, you must become familiar with DNS concepts and know how DNS is implemented. If you are not familiar with DNS, you should consult some of the excellent resources available on the Microsoft Web site, such as *http://msdn.microsoft.com/library/en-us/dns/dns_concepts.asp.*

Hierarchical Namespace

DNS uses a hierarchical namespace to locate computers. Figure 3-1 shows an example of how the namespace is organized. The root domain is represented by a period ("."). The root domain is the beginning of the DNS namespace, and the entire namespace is located underneath the root. At the next layer under the root domain are the first-level domains, including seven generic domain names (such as com, edu, mil, net, org); about 200 country abbreviations (such as ca, uk, fr, br); plus seven new domains, such as biz, info, and pro, some of which were introduced in 2001.

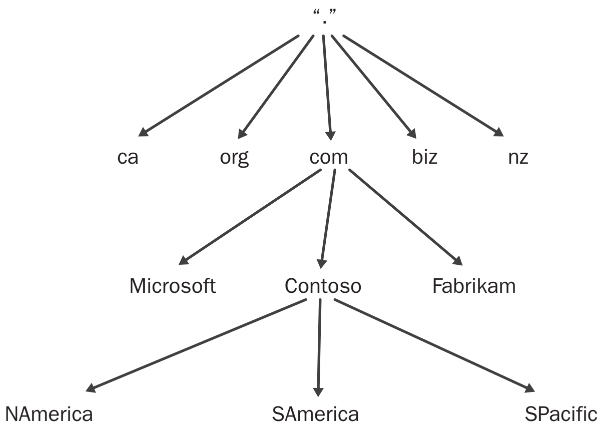


Figure 3-1. *The DNS hierarchical namespace.*

Under the top-level domains are the second-level domains, which usually refer to company names that must be registered with the Internet authority. Under the second-level domains are subdomains. Subdomains usually refer to departments or divisions within a company. These subdomains are registered and managed on the DNS servers that contain the information about the second-level domain.

Another way to think about the hierarchical namespace is to consider a fully qualified domain name (FQDN) such as www.NAmerica.Contoso.com. An FQDN is a complete name that can be used to identify a specific computer within the entire DNS namespace.

To understand how the FQDN identifies the computer in the DNS namespace, read it from right to left. At the far right is the period (".") that identifies the root domain; this is the period that precedes the first-level domain name. This is followed by the first-level com domain, the second-level Contoso domain, and the NAmerica subdomain. At the far left of the FQDN is “www,” the host name of the specific computer.

Distributed Database

Because DNS uses a hierarchical namespace, it is also easy to configure it as a *distributed database*. Before DNS was implemented on the Internet, all of the name resolution information for the Internet was stored in a single file. However, as the number of hosts on the Internet increased to thousands of computers, managing the one file became impractical. To deal with this, DNS was designed to use a distributed database.

Using a distributed database means that the DNS information is stored on many computers throughout the world (in the case of the Internet) and throughout your network (in the case of an internal network). Each DNS server maintains only one small part of the DNS database. The entire database is divided into zone files based on domain names. The zone files are distributed across multiple servers. For example, there are approximately a dozen servers that contain the zone files for the root domain. These servers contain information about the DNS servers that have the zone information for the top-level domains. The root servers do not contain all the information about the top-level domains, but they do know which servers maintain that information.

The DNS servers that contain the information about the top-level domains also contain the information about which servers contain the zone files for the next level domains. For example, a server may contain the zone files for the com domain. This means that this server knows about all of the second-level domains that are registered with the com domain, but it might not know all of the details about the second-level domain. Again, the top-level domain server knows which computers at the next level contain the details about the second-level domain. This continues all the way down the DNS namespace. The server responsible for the com domain would have Contoso registered as a valid second-level domain. This server would refer any requests for information about the Contoso domain to the server that contains the zone files for Contoso.com.

Using this method of distributing the database means that no one server on the Internet needs to contain all of the DNS information. Most servers will have information about some part of the tree, but when they receive a request they cannot fulfill, they know which DNS server has the required information. The DNS servers use delegation records, forwarders, and root hints to determine which DNS server has the required information. These topics will be discussed later in this chapter.

Name Resolution Process

The DNS hierarchical namespace and distributed database are used when a client tries to locate the IP address of an Internet resource. To use the example from the previous section (and illustrated in Figure 3-1), a DNS client (also called a resolver) anywhere in the world might want to connect to the Web server at *www.NAmerica.Contoso.com*.

➲ **To acquire the IP address, perform the following steps:**

1. The client resolver begins by sending a recursive query to its configured DNS server (usually the DNS server of an Internet Service Provider [ISP]) asking for the IP address. A *recursive query* is a query that can have only two possible answers: the IP address that the client is looking for or an error message indicating that the information cannot be found.
2. If the ISP's DNS server has the requested information in its cache, it returns the IP address to the user. If it does not, the DNS server tries to find the information by sending an iterative query to another server that might have the information. The response to an iterative query can be either the name resolution that the client requested or a referral to another DNS server that might be able to fulfill the request. In our example, the ISP's DNS server sends an iterative query to a root server asking for the IP address for www.NAmerica.Contoso.com.
3. The root server cannot supply the answer to the query, but it does reply with a list of servers responsible for the top-level com domain. This process of providing alternate DNS servers to contact is called a *referral*. The ISP's DNS server sends an iterative query to one of these servers, asking for the IP address.
4. The com server replies with a list of servers that are responsible for the Contoso.com domain. The ISP's DNS server then queries the Contoso.com DNS server, which responds with the names of the DNS servers that manage the NAmerica.Contoso.com domain.
5. The NAmerica.Contoso.com DNS server contains all of the information about this domain, so it responds to the ISP's DNS server with the IP address for the requested host.
6. The ISP's DNS server responds to the recursive query it received from the client resolver and sends the IP address for the requested Web server.
7. The client computer connects to www.NAmerica.Contoso.com.

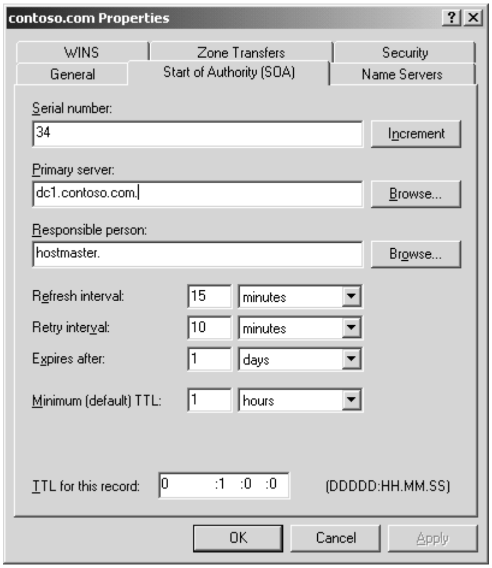
This whole process can happen very quickly, but usually not all the steps are needed. When a DNS server resolves any type of name, it saves that information in a cache for a specified period. In our example, if someone had located the same site earlier in the day and the ISP's DNS server had resolved the name, the DNS server would have looked in its cache and provided the answer immediately.

Resource Records

The actual records that are stored in the DNS zone files are called *resource records* (RRs). Resource records contain the actual information about the domain. You can create twenty-two different types of resource records on a Windows Server 2003 DNS server. The most common resource records are listed in Table 3-1.

Table 3-1. Common Resource Records in Windows Server 2003 DNS

Name	Explanation
Start of Authority (SOA)	Identifies the primary name server for the zone. Also sets parameters for the zone such as the default settings for zone transfers, expiration times on zone information, and the Time to Live (TTL). (See Figure 3-2 for an example of an SOA record.)
Host (A)	Identifies the IP address for a specific host name. This is the record that the DNS server returns during name resolution.
Mail Exchanger (MX)	Identifies Internet messaging servers. This record is used by other messaging servers on the Internet to locate the messaging servers in a domain.
Name Server (NS)	Identifies all of the name servers for the domain.
Pointer (PTR)	Identifies the host names mapped to a specific IP address. These records are stored in the reverse lookup zone.
Canonical Name (CNAME)	Identifies an alias for another host in the domain. This is used when more than one host name uses the same IP address.
Service Locator (SRV)	Identifies a service that is available in the domain. Active Directory makes extensive use of SRV records to locate domain controllers.



contoso.com Properties

Serial number: 34

Primary server: dc1.contoso.com.

Responsible person: hostmaster.

Refresh interval: 15 minutes

Retry interval: 10 minutes

Expires after: 1 days

Minimum (default) TTL: 1 hours

TTL for this record: 0 :1 :0 :0 (DDDDD:HH.MM.SS)

Figure 3-2. *An SOA record for the Contoso.com domain.*

Tip Figure 3-2 shows the SOA record in the DNS administrative tool. The DNS records can also be written in a standard text format. For example, a standard host record for a server called Web1.Contoso.com can be written as Web1.Contoso.com IN A 192.168.1.100.

DNS Domains, Zones, and Servers

One of the important aspects of learning how DNS works is understanding the terminology used to describe the DNS components.

Domains Versus Zones

One of the terminology issues that can be confusing is the difference between domains and zones. One way to understand the difference is to think of a domain as part of the DNS namespace, and a zone as being information about that part of the namespace. For example, a company may own a second-level domain name such as Contoso.com. This means that the company owns one part of the entire DNS namespace—this is their domain. When the company implements the DNS servers for the domain, all of the information about the DNS domain is stored on one or more DNS servers. This information includes all of the resource records for all of the computers in the DNS domain. This information about each domain is the *zone information*, and is stored in zone files on the DNS servers.

There are two different types of zone files in DNS: *forward lookup zones* and *reverse lookup zones*. A forward lookup zone is used primarily to resolve host names to IP addresses. The Host (A) records provide this functionality. The forward lookup zone also includes the SOA and NS records and may also include MX records, CNAME records, and SRV records. The forward lookup zone is used whenever a client resolver queries the DNS server to locate the IP address of a server on the network.

Reverse lookup zones perform the opposite function. A reverse lookup zone is used when a host's IP address is known, but the host name is not. The reverse lookup zone also has SOA and NS records, but the rest of the records are PTR records. A PTR record format is similar to a host record, but it provides the answer for a reverse lookup. (For more information on these records, see Table 3-1.)

A forward lookup zone name is the domain name. A reverse lookup zone name is more difficult to determine because it uses an IP subnet, not a domain name, as the boundary for the zone. When you create the reverse lookup zone, you must give the zone a name based on an IP subnet. For example, if you create a reverse lookup zone for the subnet 192.168.1.0, the zone name would be 1.168.192.in-addr.arpa. The in-addr.arpa is a special name reserved in the DNS namespace to refer to reverse lookup zones. The first part of the zone name is the network address, but in reverse. If you were creating a reverse lookup zone for a Class B subnet (150.38.0.0), the reverse lookup zone name would be 38.150.in-addr.arpa.

Primary Name Servers

A primary name server is the only server with a writable copy of the zone files (the zone on the primary name server is called the *primary zone*). This means that the DNS administrator must have access to the primary name server whenever any changes need to be made to the zone information. After changes have been made to the zone files, the data is automatically replicated to the secondary name servers using a process called a *zone transfer*.

Secondary Name Servers

A secondary name server has a read-only copy of the zone files. The only way the zone information on a secondary name server can be updated is through a zone transfer from the primary name server. In the early iterations of DNS, every zone transfer was a complete zone transfer, that is, the entire contents of the DNS zone file were transferred from the primary name server to the secondary server. Request for Comment (RFC) 1995 introduced a more efficient zone transfer mechanism called an *incremental zone transfer* in which only the changes made to the zone files since the last transfer are replicated to the secondary server. Another improvement to the zone transfer process is described in RFC 1996, which describes a notification mechanism that enables the primary server to alert the secondary name servers when changes have been made to the zone files. Without the notification option, the secondary name server will only contact the primary name server at the refresh intervals defined on the SOA record for each zone.

Note The Windows Server 2003 DNS server supports both incremental zone transfers and notifications. The Windows Server 2003 DNS server also supports Active Directory integrated zones where the regular zone transfer is replaced by Active Directory replication.

Caching-Only Name Servers

A third type of name server is a caching-only server. This server does not manage any zone files; rather, it only caches any name resolutions that it has completed. A caching-only server is frequently used in remote offices with a limited bandwidth connection to a larger office. Because the caching-only server does not have any zone files, there is no DNS zone transfer traffic across a slow network connection. The caching-only server should be configured to forward all DNS requests to a server at the company's main office. As the caching-only server resolves DNS requests, it caches the information for a certain period (the default is 1 hour). This means that any local DNS requests for the same information can be resolved locally.

Note All Windows Server 2003 DNS servers, including primary and secondary name servers, are caching servers, but not caching-only servers. The difference between those servers and caching-only servers is that the latter do not have any zone information.

Zones of Authority

To fully understand DNS, you must be familiar with zones of authority, or authoritative name servers. Each primary and secondary name server is authoritative for its domain. For example, if a DNS server contains the zone files for the Contoso.com domain, that server is the authoritative name server for that domain. As the authoritative name server, the server will not forward any queries about hosts in that zone to any other DNS server. Many companies set up a DNS server configuration similar to that shown in Figure 3-3. In this scenario, there are two primary DNS servers configured for the Contoso.com domain. DNS1 contains a host record for a server called Web1.Contoso.com, but DNS2 does not have this record. When a client connects to DNS1, it will be able to resolve the IP address for Web1. When a client connects to DNS2 and requests the IP address for Web1, the server will respond that the host cannot be found. Because DNS2 is authoritative for the Contoso.com domain, it will never forward the request to DNS1. Even if DNS2 has DNS1 configured as a forwarder or root hint, it will never forward requests to the other server because it is the authoritative server for the Contoso.com domain. This behavior is by design and, as the following real-world discussion shows, offers a specific security advantage.

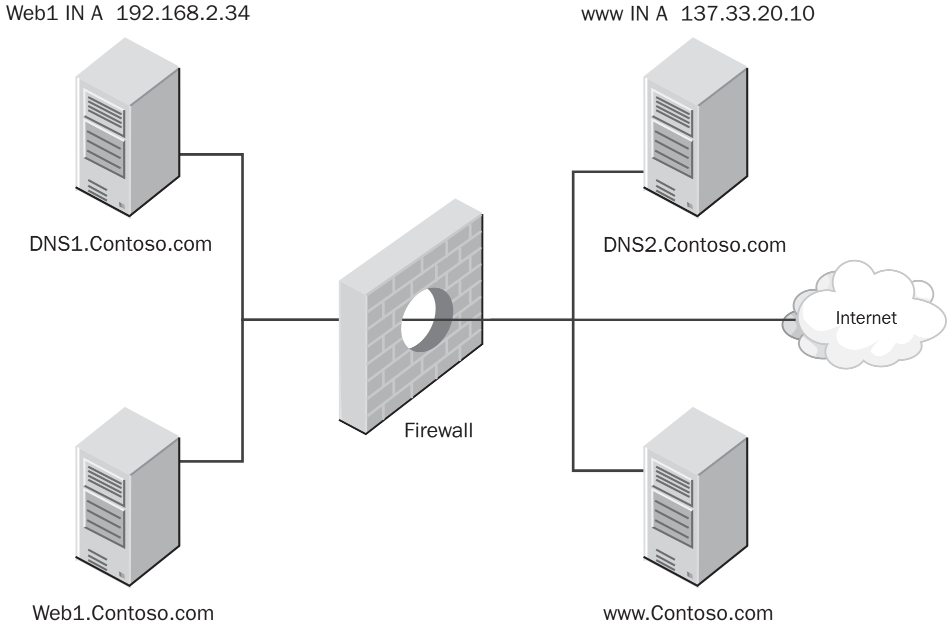


Figure 3-3. *Multiple authoritative DNS servers.*

Real World Using Multiple Authoritative Servers for the Same Zone

The most common scenario in which a company may have two DNS servers that are both authoritative for the same domain is when the company's Internet DNS name and the internal DNS name are identical (as illustrated in Figure 3-3). The DNS1 server is inside the firewall, and the DNS2 server is outside it. DNS2 is used to resolve DNS requests for Internet clients, while DNS1 is used for internal clients and for Active Directory SRV records. Because both servers are authoritative for the same zone (Contoso.com), they will never forward requests for this domain to each other.

In such a scenario, you should maintain unique zone information on each DNS server. The external DNS server is likely to have a relatively small zone file consisting of the Web servers and MX records that need to be accessible to the Internet. The internal DNS server will likely have a much larger zone file that includes all of the domain controller records, all internal server records, and possibly the host records for all the client computers on the network. The only duplication between the two zone files might be some of the external DNS records. For example, when the internal clients connect to www.Contoso.com, you might want them to connect to the same external Web server as that accessed by Internet clients. In this case, you must include the host record for the Web server in the zone file on DNS1. If you don't, the internal clients will not be able to connect to the Web server.

Delegated Zones

Since DNS uses a hierarchical namespace, there must be some way of connecting the layers of the hierarchy together. For example, if a client connects to a server that is authoritative for the com first-level domain, and requests a server in the Contoso.com domain, the com server must have some way of determining which name servers are authoritative for the Contoso.com domain. This is made possible by the use of *delegation records*.

A delegation record is a pointer to a lower-level domain that identifies the name servers for the lower-level domain. For example, as shown in Figure 3-4, DNS1.Contoso.com is an authoritative name server for the Contoso.com domain. DNS2 and DNS3 are authoritative name servers for the NAmerica.Contoso.com domain. DNS1 is considered authoritative for the NAmerica.Contoso.com domain but does not have all of the resource records for the child domain. However, DNS1 uses a delegation record pointing to DNS2 and DNS3 as the name servers for the child domain. When a client connects to DNS1 requesting information about NAmerica.Contoso.com, the server will refer the client to the name servers for the child domain.

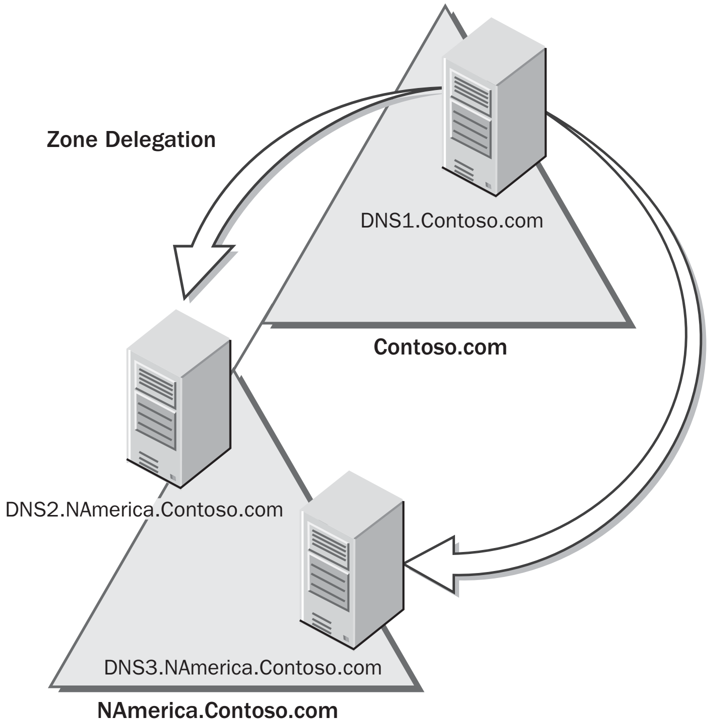


Figure 3-4. *Delegated zones.*

Forwarders and Root Hints

The second method for connecting the different layers of the DNS hierarchy together is by using forwarders and root hints. In most cases, forwarders and root hints are used by those DNS servers lower in the DNS namespace to locate information from DNS servers higher up in the hierarchy. Both forwarders and root hints are used by the DNS server to locate information that is not in its zone files. For example, a DNS server may be authoritative for only the Contoso.com domain. When this DNS server receives a query from a client requesting a name resolution in the Fabrikam.com domain (see Figure 3-1), the Contoso.com DNS server must have some way of locating this information.

One way to configure this is to use *forwarders*. A forwarder is simply another DNS server that a particular DNS server uses when it cannot resolve a query. For example, the authoritative name server for Contoso.com might receive a recursive query for the Fabrikam.com domain. If the Contoso DNS server has been configured with a forwarder, it will send a recursive query to the forwarder requesting this information. Forwarders are often used on an organization's internal network. An organization may have several DNS servers with the primary task of internal name resolution. However, users inside the organization are also likely to need to resolve Internet IP addresses. One way to enable this is to configure all the internal DNS servers to try to resolve the Internet addresses. A more common configuration has all the internal DNS servers configured with a forwarder pointing to one DNS server that is responsible for Internet name resolution. This latter configuration is shown in Figure 3-5. All the internal DNS servers forward any query for a non-authoritative zone to one DNS server, which then tries to resolve the Internet addresses. If a DNS server is configured with more than one forwarder, that DNS server will try all of the forwarders, in order, before trying any other way of resolving the IP addresses.

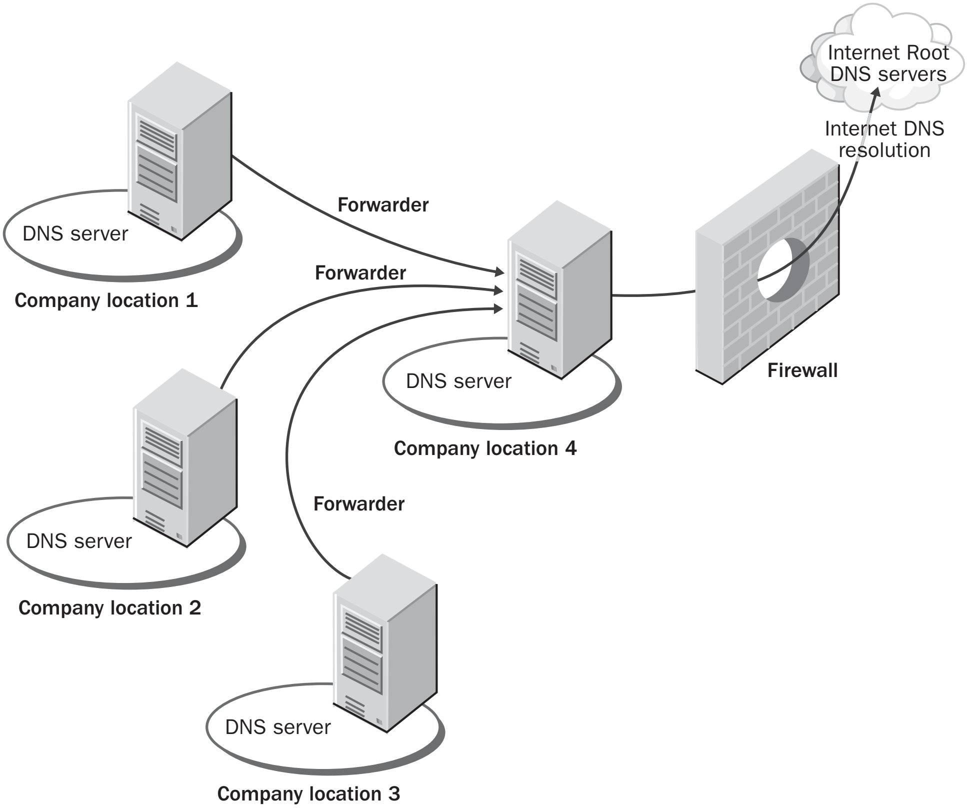


Figure 3-5. *Using forwarders for Internet name resolution.*

The second method available to a DNS server for resolving queries for zones for which it is not authoritative is the use of *root hints*. When you install a Windows Server 2003 DNS server that has access to the Internet, the server is automatically configured with a standard list of root servers. These servers are the servers that are authoritative for the root of the Internet namespace. If a DNS server receives a query for a DNS zone for which it is not authoritative, the server will send an iterative query to one of the root servers, initiating a series of iterative queries until the name is resolved or until the server has confirmed that the name cannot be resolved.

Note The root servers that are automatically configured on the DNS server are copied from the Cache.dns file that is included with the DNS server setup files. You can add additional DNS servers to the root hint list, including the DNS servers on your internal network.

By default, Windows Server 2003 DNS servers use both forwarders and root hints to try to resolve names. If the server is configured with forwarders, it will send recursive queries to all of the configured forwarders first. If none of the forwarders can provide the required information, the DNS server will then begin sending iterative queries to the servers configured as root hints. In some cases, however, you may want to have the DNS server use the forwarders only, and not use the root hints. To configure the DNS server to use the forwarders only, you must select the Do Not Use Recursion For This Domain option on the Forwarders tab of the DNS server's Properties sheet. If you select this option, the DNS server will first try to resolve any queries from its local zone information or its cached information. It will then try to resolve the queries by sending recursive queries to each of its forwarders. However, if the forwarders cannot provide the requested information, the DNS server will not use any other means to locate the information. If the DNS server cannot resolve the query using forwarders, it will inform the client that the host cannot be found.

Note Windows Server 2003 DNS has added greater functionality to the traditional forwarders by implementing conditional forwarders. This topic is covered in detail in "Conditional Forwarding," later in this chapter.

Dynamic DNS

In the past, one difficult aspect of working with DNS was that all the zone information had to be manually entered into the DNS server. Until RFC 2136, there was no way to update the DNS server zone information automatically. However, as described in RFC 2136, DNS servers can now be configured to accept automatic updates to the resource records in the zone files. This option is called dynamic DNS (DDNS).

Windows Server 2003 DNS servers support dynamic DNS. By default, all Windows 2000 and Windows XP Professional clients as well as Windows 2000 Server; Windows 2000 Advanced Server; Windows 2000 Datacenter Server; Windows Server 2003, Standard Edition; Windows Server 2003, Enterprise Edition; and Windows Server 2003, Datacenter Edition automatically update their resource records in DNS. In addition, Windows 2000 and Windows Server 2003 domain controllers automatically register SRV records with the DNS servers that are used to locate domain controllers. Windows Server 2003 DNS servers will also accept dynamic record registration from Dynamic Host Configuration Protocol (DHCP) servers. The Windows Server 2003 DHCP server can be configured to automatically update the DNS records for any of its clients, including Microsoft Windows 95, Microsoft Windows 98, Microsoft Windows Me, or Microsoft Windows NT clients.

One of the concerns with dynamic DNS is security. Without some control over who can update the DNS resource records, anyone with access to your network can potentially create a resource record in your DNS zone files and then use the record to redirect network traffic. To deal with this, Windows Server 2003 DNS provides for *secure updates*. Secure updates are only available in Active Directory integrated zones. With secure updates, you can control who has the right to register and update the DNS records. By default, the members of the Authenticated Users group have the right to update their records in DNS. However, you can change this by modifying the access control list (ACL) for the DNS zone.

Dynamic DNS greatly reduces the amount of work that the DNS administrator needs to do. As will be seen in the next section, Active Directory in Windows Server 2003 requires that the SRV records for each domain controller are listed in the zone information, so enabling dynamic updates is an important feature in Windows Server 2003 DNS.

DNS and Windows Server 2003 Active Directory

Active Directory cannot function without a reliable DNS configuration. Without DNS, the domain controllers cannot locate each other to replicate domain information, and the Windows 2000 and Windows XP Professional clients will be very slow in locating the domain controllers to log on. In addition, without a reliable DNS, any other services that require Active Directory will fail. For example, Exchange 2000 Server stores all of its configuration information in Active Directory, so if the servers running Exchange 2000 Server cannot locate a domain controller when the servers start, they will not be able to start most of the Exchange 2000 Server services.

Tip Windows 95, Windows 98, Windows ME, and Windows NT clients do not rely on DNS to locate the Windows Server 2003 domain controllers. These clients must use NetBIOS names to locate the domain controllers and usually use Windows Internet Naming Service (WINS) to resolve the NetBIOS names to IP addresses. By default, Windows Server 2003 supports these down-level clients by registering the required NetBIOS names with WINS.

DNS Locator Service

DNS is so important in Active Directory because DNS provides the information that clients need to locate the domain controllers on the network. This section takes a detailed look at the process that a client uses to locate the domain controllers.

Note In Windows NT, domain logon was based on NetBIOS names. Every domain controller registered the NetBIOS name *Domainname* with a <*1C*> as the sixteenth character in the name on the network and in WINS. When a client tried to log on to the network, the client would try to locate the servers that had the domain controller name registered. If the client could not locate one of these servers, the logon would fail. The SRV records in Windows Server 2003 are used by

Windows 2000 and Windows XP Professional clients to locate domain controllers. Without the SRV records, these clients will also not be able to log on to the Windows Server 2003 domain.

DNS Resource Records Registered by the Active Directory Domain Controller

To facilitate the location of domain controllers, Active Directory uses service locator, or SRV, records. An SRV record is a new type of DNS record described in RFC 2782, and is used to identify services located on a Transmission Control Protocol/Internet Protocol (TCP/IP) network. Every SRV record uses a standard format, as shown in the following example of one of the records used by Active Directory and explained in Table 3-2.

```
_ldap._tcp.contoso.com. 600 IN SRV 0 100 389 dc2.contoso.com
```

Table 3-2. The SRV Record Components

Component	Example	Explanation
Service	_ldap	The service that this record identifies. Additional services include _kerberos, _kpassword, and _gc.
Protocol	_tcp	The protocol used for this service. Can be either TCP or user datagram protocol (UDP).
Name	contoso.com	The domain name that this record refers to.
TTL	600	The default Time to Live for this record (in seconds).
Class	IN	The standard DNS Internet class.
Resource Record	SRV	Identifies the record as an SRV record.
Priority	0	Identifies the priority of this record for the client. If multiple SRV records exist for the same service, the clients will try to connect first to the server with the lowest priority value.
Weight	100	A load balancing mechanism. If multiple SRV records exist for the same service and the priority is identical for all the records, clients will choose the records with the higher weights more often.
Port	389	The port used by this service.
Target	dc2.contoso.com	The host that provides the service identified by this record.

Essentially, the information in this record says that if a client is looking for a Lightweight Directory Access Protocol (LDAP) server in the Contoso.com domain, the client should connect to dc2.contoso.com.

The domain controllers in a Windows Server 2003 domain register many SRV records in DNS. The following list includes all of the records registered by the first server in a forest.

```
contoso.com. 600 IN A 192.168.1.201
_ldap._tcp.contoso.com. 600 IN SRV 0 100 389 dc2.contoso.com.
_ldap._tcp.Default-First-Site-Name._sites.contoso.com. 600 IN SRV 0 100 389
    dc2.contoso.com.
_ldap._tcp.pdc._msdcs.contoso.com. 600 IN SRV 0 100 389 dc2.contoso.com.
_ldap._tcp.gc._msdcs.contoso.com. 600 IN SRV 0 100 3268 dc2.contoso.com.
_ldap._tcp.Default-First-Site-Name._sites.gc._msdcs.contoso.com. 600 IN SRV 0
    100 3268 dc2.contoso.com.
_ldap._tcp.64c228cd-5f07-4606-b843-d4fd114264b7.domains._msdcs.contoso.com.
    600 IN SRV 0 100 389 dc2.contoso.com.
gc._msdcs.contoso.com. 600 IN A 192.168.1.201
    175170ad-0263-439f-bb4c-89eacc410ab1._msdcs.contoso.com. 600 IN CNAME
    dc2.contoso.com.
_kerberos._tcp.dc._msdcs.contoso.com. 600 IN SRV 0 100 88 dc2.contoso.com.
_kerberos._tcp.Default-First-Site-Name._sites.dc._msdcs.contoso.com. 600 IN
    SRV 0 100 88 dc2.contoso.com.
_ldap._tcp.dc._msdcs.contoso.com. 600 IN SRV 0 100 389 dc2.contoso.com.
_ldap._tcp.Default-First-Site-Name._sites.dc._msdcs.contoso.com. 600 IN SRV 0
    100 389 dc2.contoso.com.
_kerberos._tcp.contoso.com. 600 IN SRV 0 100 88 dc2.contoso.com.
_kerberos._tcp.Default-First-Site-Name._sites.contoso.com. 600 IN SRV 0 100 88
    dc2.contoso.com.
_gc._tcp.contoso.com. 600 IN SRV 0 100 3268 dc2.contoso.com.
_gc._tcp.Default-First-Site-Name._sites.contoso.com. 600 IN SRV 0 100 3268
    dc2.contoso.com.
_kerberos._udp.contoso.com. 600 IN SRV 0 100 88 dc2.contoso.com.
_kpasswd._tcp.contoso.com. 600 IN SRV 0 100 464 dc2.contoso.com.
_kpasswd._udp.contoso.com. 600 IN SRV 0 100 464 dc2.contoso.com.
DomainDnsZones.contoso.com. 600 IN A 192.168.1.201
_ldap._tcp.DomainDnsZones.contoso.com. 600 IN SRV 0 100 389 dc2.contoso.com.
_ldap._tcp.Default-First-Site-Name._sites.DomainDnsZones.contoso.com. 600 IN
    SRV 0 100 389 dc2.contoso.com.
ForestDnsZones.contoso.com. 600 IN A 192.168.1.201
_ldap._tcp.ForestDnsZones.contoso.com. 600 IN SRV 0 100 389 dc2.contoso.com.
_ldap._tcp.Default-First-Site-Name._sites.ForestDnsZones.contoso.com. 600 IN
    SRV 0 100 389 dc2.contoso.com.
```

Note When one of the Windows Server 2003 servers is promoted to a domain controller, all of these records are written to a file called Netlogon.dns, which is located in the %systemroot%\system32\config folder. If you do not want to enable dynamic updates on the DNS servers, you can import these records into the DNS zone files.

The first part of the SRV record identifies the service that the SRV record points to. The possible services are:

- **_ldap** Active Directory is an LDAP-compliant directory service, with the domain controllers operating as LDAP servers. The _ldap SRV records identify the available LDAP servers on the network. These servers could be Windows Server 2003 domain controllers or other LDAP servers.

- **_kerberos** The primary authentication protocol for all Windows 2000 and Windows XP Professional clients. The _kerberos SRV records identify all the Key Distribution Centers (KDCs) on the network. These could be Windows Server 2003 domain controllers or other KDC servers.
- **_kpassword** The _kpassword SRV record identifies the kerberos password-change servers on the network (again either Windows Server 2003 domain controllers or other kerberos password-change servers).
- **_gc** The _gc SRV record is specific to the global catalog function in Active Directory. The global catalog server serves a number of important functions in Active Directory.

Many of the SRV records also contain a site identifier in addition to the components listed in Table 3-2. A site is used in Active Directory to identify one or more IP subnets that are connected with fast network connections. One of the advantages of using sites is that the network clients will always try to log on to a domain controller that resides in the same site as the client. The site records are essential for the computers to locate domain controllers in the same site as the client. The exact process that a client uses to locate the site information is discussed in the next section.

Another essential component of the SRV records is the *_msdcs* value that appears in many of the records. Some of the services provided by the SRV records are non–Microsoft specific. For example, there could be non-Microsoft implementations of LDAP or kerberos servers on the network. These servers could also register an SRV record with the DNS server. Windows Server 2003 domain controllers register the generic records (for example, _ldap._tcp.contoso.com), but the domain controllers also register records containing the _msdcs reference. These records refer only to Microsoft-specific roles, that is, to Windows Server 2003 or Windows 2000 domain controllers. The records identify the primary function of each server as gc (global catalog), dc (domain controller) or pdc (primary domain controller emulator).

Another record that is registered contains the domain's globally unique identifier (GUID). The domain GUID record is used to locate domain controllers in the event of a domain rename.

Note There are also several records included under the ForestDnsZones and the DomainDnsZones subdomains. These records are discussed in more detail in "Application Directory Partitions," later in this chapter.

Active Directory Domain Controller Location Process

The domain controllers running Windows Server 2003 register some or all of the records, described earlier, in DNS. These records then play an essential role when a client like Windows 2000 or Windows XP Professional tries to log on to the domain. The following steps describe the process that these clients use to log on to the domain.

1. When the user logs on, the client computer sends a remote procedure call (RPC) to the local Net Logon service initiating a logon session. As part of the RPC, the client sends information such as the computer name, domain name, and site name to the Net Logon service.
2. The Net Logon service uses the domain locator service to call the DsGetDcName() API, passing one of the flag parameter values listed in Table 3-3.

Table 3-3. A Subset of the DsGetDcName Flag Parameter Values

DsGetDcName flag values	DNS record requested
DS_PDC_REQUIRED	_ldap._tcp.pdc._msdcs.*domainname*
DS_GC_SERVER_REQUIRED	_ldap._tcp.*sitename*._sites.gc._msdcs.*forestrootdomainname*
DS_KDC_REQUIRED	_kdc._tcp.*sitename*._sites.dc._msdcs.*domainname*
DS_ONLY_LDAP_NEEDED	_ldap._tcp.*sitename*._sites._msdcs.*domainname*

> **Note** In almost all cases, the DsGetDcName function also includes the *sitename* parameter. For all of the requests except the DS_PDC_REQUIRED request, the client always makes an initial request using the site parameter. If the DNS server does not respond to the request, the client will send the same request without the site parameter. For example, if the DS_KDC_REQUIRED request is not fulfilled, the client will send a request for the _kdc._tcp.dc._msdcs.*forestrootdomain* record. This can happen when the client is in a site that is not recognized by the DNS servers.
>
> The client may also pass the *DomainGUID* parameter rather than the domain name to DsGetDcName(). In this case, the client is requesting the _ldap._tcp.*domainGUID*.domains._msdcs.*forestname* record. This will only happen when a domain has been renamed.

3. The DNS server returns the requested list of servers, sorted according to priority and weight. The client then sends an LDAP query using UDP port 389 to each of the addresses in the order they were returned. After each packet is sent, the client waits for 0.1 second and if no response is received, it sends a packet to the next domain controller. The client continues this process until it receives a valid response or has tried all of the domain controllers.
4. When a domain controller responds to the client, the client checks the response to make sure that it contains the requested information. If it does, the client begins the logon process with the domain controller.
5. The client caches the domain controller information so that the next time it needs to access Active Directory it does not have to go through the discovery process again.

How the Client Determines Which Site It Belongs To

Having site-specific records is important in order for Active Directory to operate efficiently, because a lot of client activity is limited to a particular site. For example, the client logon process always tries to connect to a domain controller in the client site before connecting to any other sites. So how does the client know which site it belongs to?

The site information for the forest is stored in the configuration directory partition in Active Directory, and this information is replicated to all domain controllers in the forest. Included with the configuration information is a list of IP subnets that are associated with a particular site. When the client logs on to Active Directory for the first time, the first domain controller to respond compares the client's IP address with the site IP addresses. Part of the domain controller's response to the client is the site information, which the client then caches. Any future logon attempts will include the client site information.

If the client is moved between sites (for example, a portable computer may be connected to a network in a different city), the client still sends the site information as part of the logon. The DNS server will respond with the record of a domain controller that is in the requested site. However, if the domain controller determines that the client is not in the original site based on the client's new IP address, it will send the new site information to the client. The client then caches this information and tries to locate a domain controller on the correct subnet.

If the client is not in any site that is defined in Active Directory, it cannot make site-specific requests for domain controllers.

Active Directory Integrated Zones

One of the greatest advantages of running DNS in a Windows Server 2003 operating system is the option to use Active Directory integrated zones. Active Directory integrated zones provide a number of advantages:

- The zone information is no longer stored in zone files on the DNS server hard disk, but is stored in the Active Directory database. This provides additional security.
- The zone transfer process is replaced by Active Directory replication. Because the zone information is stored in Active Directory, the data is replicated through the normal Active Directory replication process. This means that the replication occurs at a per-attribute level so that only the changes to the zone information are replicated. Between sites, the replication traffic can also be highly compressed, saving additional bandwidth. Using an Active Directory integrated zone also enables the use of application partitions that can be used to fine-tune the replication of DNS information.
- Integrated zones offer the possibility of a multimaster DNS server configuration. Without Active Directory, DNS can support only one primary name server for each domain. That means that all changes to the zone information must be made on the primary name server and then transferred to the secondary name servers.

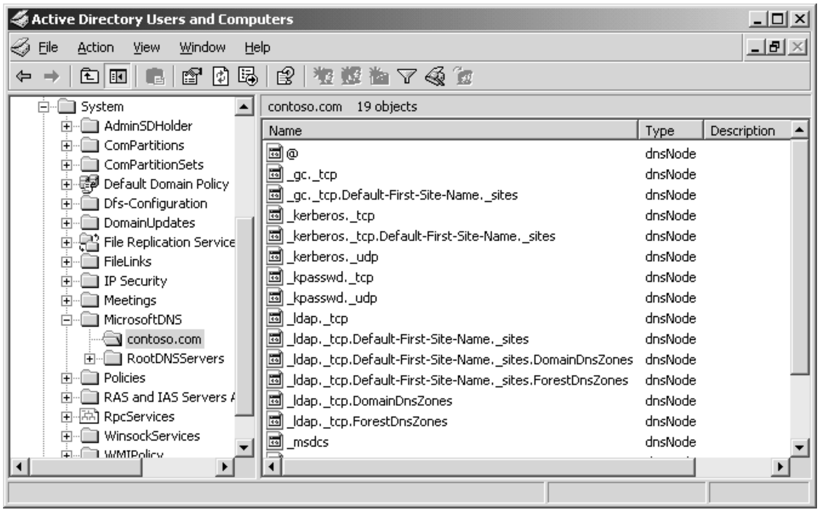
With Active Directory integrated zones, each DNS server has a writable copy of the domain information, so that changes to the zone information can be made anywhere in the organization. The information is then replicated to all other DNS servers.

- Integrated zones offer the option to enable secure updates. If a zone is configured as an Active Directory integrated zone, you can configure the zone to use secure updates only. This means that you have more control over which users and computers can update the resource records in Active Directory.

The greatest disadvantage of Active Directory integrated zones is the fact that DNS must be installed on a Windows Server 2003 domain controller. This can create an additional load on that domain controller.

Tip You can combine Active Directory integrated zones with secondary zones. For example, you might have three domain controllers in a central location with several remote offices where you do not have a domain controller. If you want to install a DNS server into a remote office, you can install DNS on a member server running Windows Server 2003 and then configure a secondary zone on the DNS server. The secondary server will then accept zone transfers from the Active Directory integrated zone.

When the zone is configured as an Active Directory integrated zone, you can view the DNS information in Active Directory (see Figure 3-6). To do this, start the Microsoft Management Console (MMC) and ensure the Active Directory Users And Computers snap-in has been added to the console. Select the Active Directory Users And Computers folder and, from the View menu, select Advanced Features. Open the folder bearing the domain name, open the System folder, and then open the MicrosoftDNS folder. The zone information for all Active Directory integrated zones is listed in each zone folder.



contoso.com 19 objects

Name	Type	Description
@	dnsNode	
_gc._tcp	dnsNode	
_gc._tcp.Default-First-Site-Name._sites	dnsNode	
_kerberos._tcp	dnsNode	
_kerberos._tcp.Default-First-Site-Name._sites	dnsNode	
_kerberos._udp	dnsNode	
_kpasswd._tcp	dnsNode	
_kpasswd._udp	dnsNode	
_ldap._tcp	dnsNode	
_ldap._tcp.Default-First-Site-Name._sites	dnsNode	
_ldap._tcp.Default-First-Site-Name._sites.DomainDnsZones	dnsNode	
_ldap._tcp.Default-First-Site-Name._sites.ForestDnsZones	dnsNode	
_ldap._tcp.DomainDnsZones	dnsNode	
_ldap._tcp.ForestDnsZones	dnsNode	
_msdcs	dnsNode	

Figure 3-6. *Viewing Active Directory integrated zone information.*

Real World DNS Resolution Without the Windows Server 2003 DNS Enhancements

A corporation consisting of four distinct companies, all of which had been independent companies before a series of acquisitions and mergers created one large corporation, was planning and deploying Active Directory in Windows 2000 Advanced Server. Each company insisted on maintaining a distinct namespace within one forest, which meant that a forest with a dedicated root domain and four company domains had to be deployed (see Figure 3-7 for an illustration of the forest plan). Each company was located in a different city in Canada.

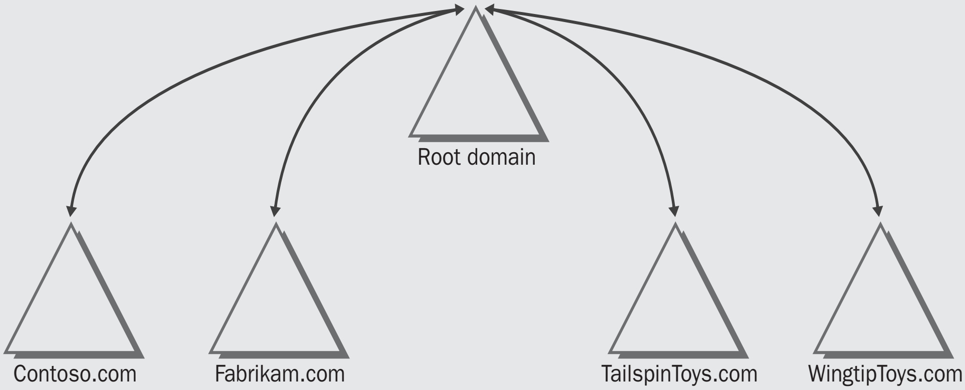


Figure 3-7. *A multiple-tree Active Directory forest design.*

Because there was no contiguous namespace for the company, the normal process of delegating child domains did not apply. The normal process of configuring forwarders and root hints was also not efficient because the forwarder and root hint configuration was not robust enough. For example, if someone from Contoso.com had needed to locate a resource in Fabrikam.com's domain, the client would have queried Contoso's DNS server. Because the server was not authoritative for the Fabrikam domain, it would have had to try to resolve the name by using a forwarder or root hint. If the Contoso DNS server was configured with a forwarder to the DNS server in the Fabrikam domain, the name could have been resolved. However, that forwarder record could not have been used to resolve computer names in the TailspinToys.com domain, or on the Internet. Using Windows 2000 DNS provides some answers to this problem, as described in the following list, but none of them are particularly satisfying.

- Creating a secondary zone on each DNS server for each of the other domains would result in too much zone transfer traffic.
- Creating a secondary zone for each company domain on the DNS servers in the root domain and configuring all of the domain DNS servers to forward requests to the root domain DNS servers would have created a significant load on the root domain DNS servers. Also, because all of the root domain DNS servers were located in one datacenter, name resolution would result in significant network traffic from other offices.

Neither of these solutions is ideal. The Windows Server 2003 operating system provides three better ways to deal with this scenario. They are *conditional forwarding*, *stub zones*, and *application directory partitions*.

DNS Enhancements

Most of the DNS options that have been discussed up to this point are available in Windows 2000. However, the Windows Server 2003 operating system has added at least three significant enhancements to DNS. The real-world scenario on the previous page illustrates one of the common difficulties in configuring DNS in a large corporation before the development of the Windows Server 2003 operating system.

Conditional Forwarding

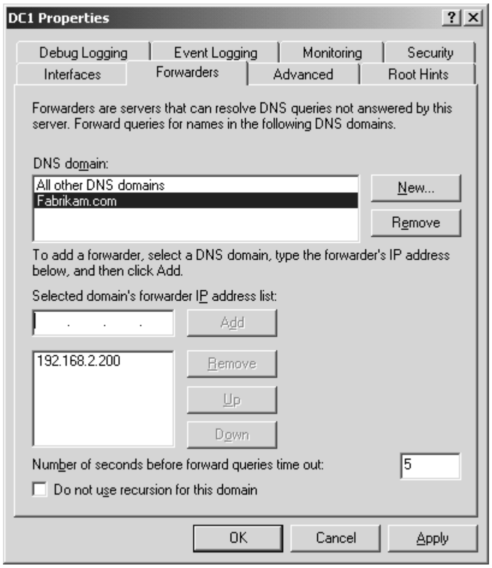
Conditional forwarding is designed to add intelligence to the forwarding process. Until the introduction of the Windows Server 2003 operating system, the forwarding process could not make any distinctions based on domain names. When a client resolver made a request that the server could not answer from its cache or zone files, the server would send a recursive query to the list of configured forwarders. There was no option to configure the forwarder to be domain-specific. Conditional forwarding provides exactly this type of intelligence: the DNS server can now forward domain requests to different DNS servers based on domain names.

For example, the company in the scenario described earlier has five trees in a single forest. To replicate, the domain controllers must be able to locate domain controllers in the other domains. Users also frequently travel between companies. They must be able to log on to their home domain regardless of which network they are physically connected to. There is also a significant amount of resource-sharing configured between the companies. These requirements mean that the DNS information must be shared across the domains.

With Windows Server 2003, the DNS servers in each domain can be configured with a conditional forwarder to one or more DNS servers in the other domains. This means that when one of the DNS servers needs to resolve a name in a different domain, it can use just the forwarder that is configured for that domain. For example, when a client in the Contoso.com domain needs to locate a resource in the Fabrikam.com domain, it queries the DNS server in the Contoso.com domain. The DNS server checks its zone files to determine if it is authoritative for the domain and then checks its cache. If it cannot resolve the name from these sources, it will check the forwarder list. One of the forwarders is specific for the Fabrikam.com domain, so the Contoso.com DNS server will send the recursive query only to that DNS server. If there are no conditional forwarders for the Fabrikam.com domain configured on the Contoso.com DNS server, it will forward the request to any forwarder that is configured without any specific domain settings and then try the root hints.

Caution Notice that the DNS server still checks its own zone files first before checking for forwarders. If a DNS server is authoritative for the domain, it will not forward the request to a conditional forwarder.

Conditional forwarding is configured on the server's Properties sheet in the DNS administrative tool (shown in Figure 3-8). Using this interface, you can configure one or more domain controllers as forwarders for each domain name. If you configure multiple DNS servers for a domain name, the DNS server will try the first DNS server on the list. If this server does not respond within the time-out value set on the Forwarders tab, the server will try the next DNS server on the list, until all of the DNS servers have been tried. If there is no conditional forwarder configured for a domain name, the server will try the DNS servers represented by All Other DNS Domains.



Forwarders are servers that can resolve DNS queries not answered by this server. Forward queries for names in the following DNS domains.

DNS domain:

All other DNS domains
Fabrikam.com

To add a forwarder, select a DNS domain, type the forwarder's IP address below, and then click Add.

Selected domain's forwarder IP address list:

192.168.2.200

Number of seconds before forward queries time out: 5

☐ Do not use recursion for this domain

Figure 3-8. *Configuring conditional forwarders.*

The DNS server will always try to match the most qualified domain name when using conditional forwarding. For example, if you have a conditional forwarder configured for Fabrikam.com and for Europe.Fabrikam.com, and a client makes a request for a server such as Web1.Europe.Fabrikam.com, the DNS server will forward the request to the DNS server for Europe.Fabrikam.com.

Stub Zones

Stub zones are the second enhancement to DNS in Windows Server 2003, and are designed to simplify the configuration of name resolution across multiple namespaces. A stub zone is similar to a secondary zone. When you set up a stub zone, you must specify the IP address of a primary name server for the zone. The server holding the stub zone then requests a zone transfer from the primary name server. What is different, however, is that the stub zone contains only the SOA records, the NS records, and the host (A) records for the name servers for the domain, rather than all of the records in the zone.

This enhances name resolution across namespaces without secondary name servers having to be used. When a DNS server is configured with a stub zone, it is not authoritative for the domain. Rather, it is just much more efficient at locating the authoritative name server for the specified zone. With stub zones, the DNS server can locate the authoritative name servers for a zone without having to contact the root hint servers. Consider how a stub zone would work in a forest with a single tree, that is, with a contiguous namespace (see Figure 3-9). Without stub zones, if a client from NAmerica.Contoso.com requests an IP address for a host in the SAmerica.Contoso.com domain, the DNS server at NAmerica.Contoso.com checks its zone files, cache, and forwarders, and if none of these sources provides the information, it sends an iterative query to a root-hint server. In this case, a DNS server in the Contoso.com root domain should be configured as a root server, so the NAmerica.Contoso.com DNS server would send the query to this root server. This root server checks its delegation records and forwards the IP address of the authoritative name servers in the SAmerica.Contoso.com domain to the NAmerica.Contoso.com name server. The NAmerica.Contoso.com name server then queries one of the SAmerica.Contoso.com DNS servers for the IP address of the server that the client requested.

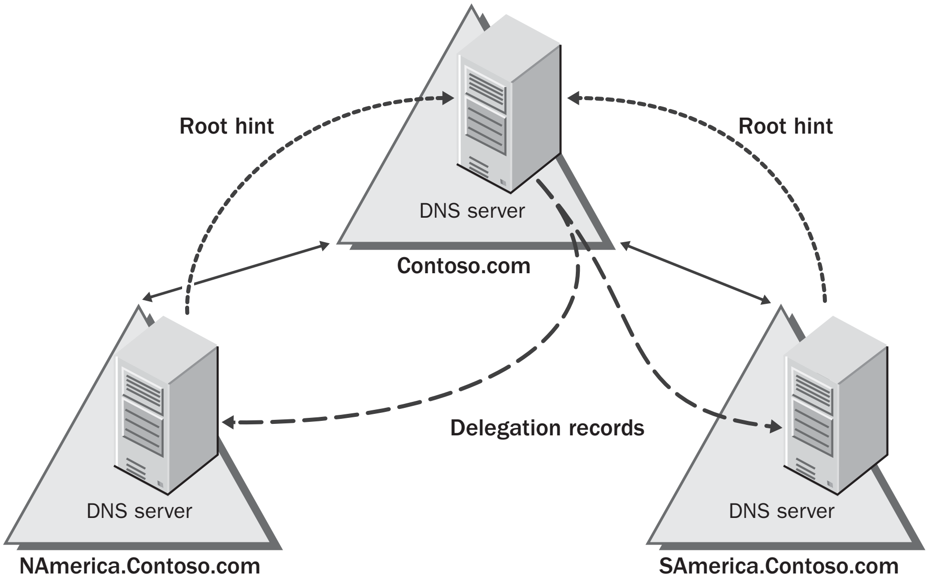


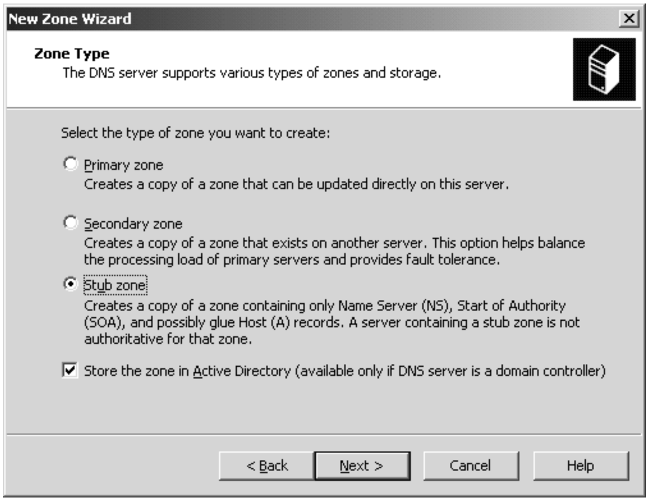
Figure 3-9. *A single forest DNS configuration without using stub zones.*

With a stub zone, the DNS server in the NAmerica.Contoso.com domain does not need to connect to the DNS server in the root domain. That is, it does not need to use its root hints to locate the name servers for the SAmerica.Contoso.com domain. Rather, when the client makes the query, the server checks its zone files, locates the stub zone, and sends an iterative query to any of the name servers in the SAmerica.Contoso.com domain.

While using stub zones in a single-tree forest may by useful (especially if there are many levels in the tree), they are even more useful in a multiple-tree scenario. Consider the previous example in which there are five trees in the forest. Using delegation records in the root zone does not work in this case because the domains do not share a common namespace. In this scenario, you can configure a stub zone for each domain on the DNS servers in the other domains. Then when any DNS query needs information from a different domain, the DNS server can use the stub zone information to immediately connect to the correct name server in the other domain.

Another useful function for stub zones is to maintain the name server list for delegated zones. When you set up a delegated subdomain, you must enter the IP address of all the name servers in the delegated domain. If that list of name servers changes—for example, if one of the name servers is removed from the network—you must manually update the delegation record. You can use a stub zone to automate the process of keeping the name server list updated. To configure this in the Contoso.com domain, you would configure a stub zone for the NAmerica.Contoso.com domain on the DNS servers in the Contoso.com domain. You would also configure a delegation record in the Contoso.com zone pointing to the stub zone. As name server records are modified in the child domain, they will be updated automatically in the stub zone. When the Contoso.com DNS servers use the delegation record, they will be referred to the stub zone, so they will always have access to the updated name server information.

To configure a stub zone, use the New Zone Wizard in the DNS administrative tool. Right-click Forward Lookup Zones (or Reverse Lookup Zones) and select New Zone. You are given the option to create a stub zone (see Figure 3-10).



Zone Type

The DNS server supports various types of zones and storage.

Select the type of zone you want to create:

○ Primary zone
Creates a copy of a zone that can be updated directly on this server.

○ Secondary zone
Creates a copy of a zone that exists on another server. This option helps balance the processing load of primary servers and provides fault tolerance.

● Stub zone
Creates a copy of a zone containing only Name Server (NS), Start of Authority (SOA), and possibly glue Host (A) records. A server containing a stub zone is not authoritative for that zone.

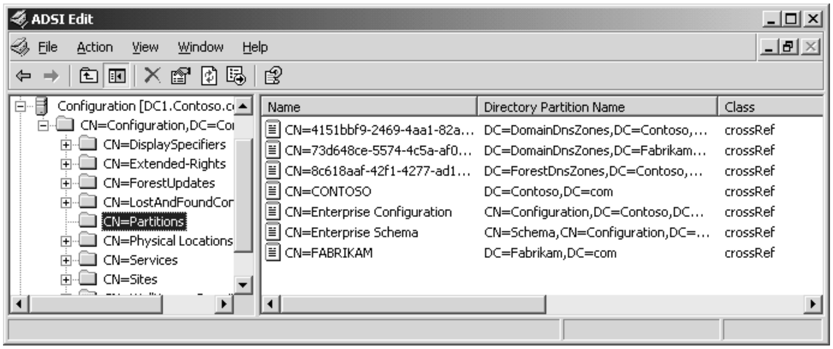
☑ Store the zone in Active Directory (available only if DNS server is a domain controller)

Figure 3-10. *Creating a new stub zone.*

Application Directory Partitions

A third enhancement to DNS that helps with resolving host names across multiple domains is the use of application directory partitions.

DNS in Windows Server 2003 Active Directory makes use of application directory partitions to facilitate the replication of DNS information throughout a forest. When you install DNS while you are promoting the first server in the forest to be a domain controller, two new directory partitions are created in Active Directory. These partitions are the DomainDnsZones partition and the ForestDnsZones partition. (These partitions are not visible in any of the regular Active Directory management tools but can be seen when using ADSI Edit or Ldp.exe; the use of ADSI Edit is shown in Figure 3-11.) Each of these partitions has a different replication configuration. The DomainDnsZones partition is replicated to all DNS servers running on domain controllers in a domain. The ForestDnsZones is replicated to all DNS servers running on domain controllers in the forest. You can also store the DNS information in the domain directory partition, which means that the DNS information will be replicated to all domain controllers in the domain.



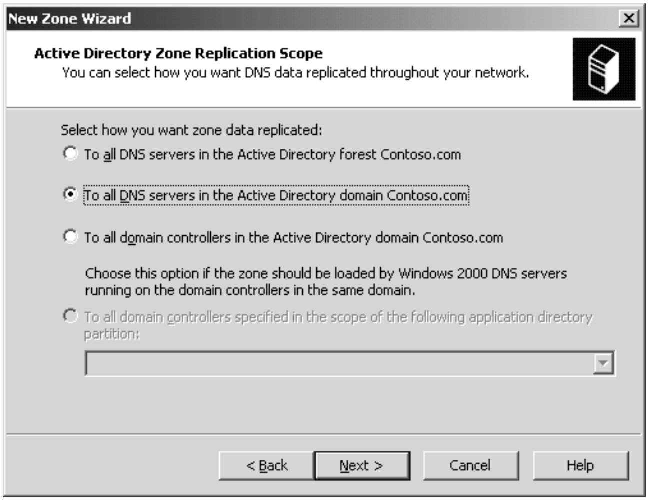
Name	Directory Partition Name	Class
CN=4151bbf9-2469-4aa1-82a...	DC=DomainDnsZones,DC=Contoso,...	crossRef
CN=73d648ce-5574-4c5a-af0...	DC=DomainDnsZones,DC=Fabrikam...	crossRef
CN=8c618aaf-42f1-4277-ad1...	DC=ForestDnsZones,DC=Contoso,...	crossRef
CN=CONTOSO	DC=Contoso,DC=com	crossRef
CN=Enterprise Configuration	CN=Configuration,DC=Contoso,DC...	crossRef
CN=Enterprise Schema	CN=Schema,CN=Configuration,DC=...	crossRef
CN=FABRIKAM	DC=Fabrikam,DC=com	crossRef

Figure 3-11. *The DNS application directory partitions in ADSI Edit.*

You are given a choice about where to store the DNS information when you create a new zone (see Figure 3-12) or through the Zone Properties sheet in the DNS administrative tool. You are given the following four choices of where to store the DNS information:

- **To All DNS Servers In The Active Directory Forest *domainname*** The information is stored in the ForestDnsZones partition, where it is replicated to all DNS servers running on domain controllers in the forest. This is the default configuration for the _msdcs zone in an Active Directory integrated zone.
- **To All DNS Servers In The Active Directory Domain *domainname*** The information is stored in the DomainDnsZones partition, to all the DNS servers running on domain controllers in the domain. This is the default configuration for the Active Directory integrated zones created during the domain controller upgrade process.

- **To All Domain Controllers In The Active Directory Domain *domainname*** The information is stored in the domain directory partition, where it is replicated to all domain controllers in the domain. The difference between this option and the option to store the information in the DomainDnsZones partition is that, in this case, all domain controllers will receive the information while the DomainDnsZones partition is only replicated to domain controllers that are also DNS servers.
- **To All Domain Controllers Specified In The Scope Of The Following Application Directory Partition** This option is only available if you create an additional application directory partition with its own replication configuration. The DNS information will be replicated to all domain controllers that have a replica of this partition.



New Zone Wizard

Active Directory Zone Replication Scope

You can select how you want DNS data replicated throughout your network.

Select how you want zone data replicated:

○ To all DNS servers in the Active Directory forest Contoso.com

● To all DNS servers in the Active Directory domain Contoso.com

○ To all domain controllers in the Active Directory domain Contoso.com

Choose this option if the zone should be loaded by Windows 2000 DNS servers running on the domain controllers in the same domain.

○ To all domain controllers specified in the scope of the following application directory partition:

Figure 3-12. *Configuring the replication scope for DNS zones.*

Note The DNS application directory partitions are created only if you choose to install DNS when you promote the first domain controller in the domain or forest. If you want to take advantage of DNS application directory partitions after you have already upgraded the domain controller, you must manually create the partition before you can use them. To create the partitions, you can use the DNS administrative tool or the DNSCMD command-line tool. If you are using the DNS administrative tool, right-click on the DNS server name and select Create Default Application Directory Partitions. If you are using DNSCMD, open a command line and type *dnscmd* ***DNSservername*** */CreateBuiltinDirectoryPartitions /forest*. This will create the ForestDnsZones partition. To create the DomainDnsZones partition, use "*/domain*" as the last parameter in the command, instead of "*/forest*". Because this command modifies the configuration directory partition in Active Directory, you must be logged in as a member of the Enterprise Admins group.

Normally, you should not change the default configuration for the zones. If you have multiple domain controllers in a domain but only some of them are DNS servers, using the DomainDnsZones partition reduces the amount of replication to the domain controllers that are not DNS servers. The _msdcs zone for each domain, which includes only information about the Active Directory servers in the domain, is stored in the ForestDnsZones partition. This information is replicated throughout the forest, so the DNS information needed to locate all the domain controllers in the forest is replicated to all DNS servers in the forest.

Summary

DNS is an essential network service for Windows Server 2003 networks. Without a stable DNS infrastructure, almost all logon and resource location efforts will fail on a Windows Server 2003 network. As a network administrator for a Windows Server 2003 network, you must become a DNS expert. This chapter provided an overview of how DNS works as a network service in any environment. It then discussed specifically the integration of DNS with Active Directory. The most important component of the integration is the domain controller locator process where the domain controllers in Active Directory register SRV records in DNS, and then the clients use these records to locate the domain controllers. Also covered were some of the DNS enhancements that Windows Server 2003 provides.