Windows 2000 Server

Chapter 3 - Name Resolution in Active Directory

Finding information in Active Directory™, the Microsoft® Windows® 2000 directory service, involves first locating an Active Directory server (domain controller) for logging on to a domain and then finding the information that you need in Active Directory. Both processes use name resolution. When you are locating a domain controller, the Domain Name System (DNS) resolves (by DNS name resolution) a domain name or computer name to an Internet Protocol (IP) address. During the search for information in Active Directory, Windows 2000 resolves (by Lightweight Directory Access Protocol [LDAP] name resolution) a distinguished name to a domain controller that holds the entry for that name.

In This Chapter

Related Information in the Resource Kit

- For more information about DNS, Transmission Control Protocol/Internet Protocol (TCP/IP) networks, subnets, and subnet masks, see the *Microsoft® Windows® 2000 Server Resource Kit TCP/IP Core Networking Guide*.
- For information about planning and deploying of domains and domain controllers, see the *Microsoft® Windows® 2000 Server Resource Kit Deployment Planning Guide*.

Locating Active Directory Servers

When an application requests access to Active Directory, an Active Directory server (domain controller) is located by a mechanism called the domain controller locator (Locator*)*. Locator is an algorithm that runs in the context of the Net Logon service. Locator can find domain controllers by using DNS names (for IP/DNS-compatible computers) or by using Network Basic Input/Output System (NetBIOS) names (for computers that are running Microsoft® Windows® version 3.*x*, Microsoft® Windows® for Workgroups, Microsoft® Windows NT® version 3.5 or later, Microsoft® Windows® 95, Microsoft® Windows® 98, or for computers on a network where IP transport is not available).

Note In this chapter, the term "Windows NT 4.0-compatible Locator" refers to the locator process that is used by clients that are running Windows 3.*x*, Windows for Workgroups, Windows NT 3.5 or later, Windows 95, or Windows 98 to locate a domain controller in any domain, as well as by clients that are running Windows 2000 to locate a domain controller that is in either a Microsoft® Windows NT® version 3.51 domain or Microsoft® Windows NT® version 4.0 domain.

The focus of this chapter is on the process of locating a domain controller. For more information about DNS resolution of host IP addresses, see "Introduction to DNS" in the *TCP/IP Core Networking Guide*.

Domain Controller Name Registration

Every Windows 2000-based domain controller registers two types of names at startup:

- A DNS domain name with the DNS service (for example, noam.reskit.com).
- A NetBIOS name with Windows Internet Name Service (WINS) or another transport-specific service (for example, noam).

When a user starts a computer and logs on to a domain, the computer must do one of two things:

- If the name of the logon domain is a DNS name, the computer must query DNS to find a domain controller with which to authenticate.
- If the name of the logon domain is a NetBIOS name, the computer must send a mailslot message to find a domain controller for the specified domain.

After the computer has found a domain controller, the information is cached so that a new query is not required for subsequent logon sessions.

DNS Domain Name Registration

Active Directory supports dynamic registration of domain controller addresses in DNS. After Active Directory has been installed during domain controller creation, the Net Logon service dynamically creates records in the DNS database that are used to locate the server. Dynamic update (described in Request for Comments (RFC) 2136) is a recent addition to the DNS standard; this addition to the standard defines a protocol for dynamically updating a DNS server with new or changed resource record values. Before the advent of this new protocol, administrators had to manually create the records that are stored on DNS servers. The implementation of DNS server that is included with Windows 2000 supports dynamic updates, as does the Berkeley Internet Name Domain (BIND) version 8.*x* implementation of DNS. (For more information about BIND DNS, see "Windows 2000 DNS" in the *TCP/IP Core Networking Guide*.)

Every Windows 2000-based domain controller dynamically registers service records (SRV records) in DNS, which allow servers to be located by service type (for example, LDAP) and protocol (for example, Transmission Control Protocol [TCP]). Because domain controllers are LDAP servers that communicate over TCP, SRV records can be used to find the DNS computer names of domain controllers. In addition to registering LDAP-specific SRV records, Net Logon also registers Kerberos v5 authentication protocol-specific SRV records to enable locating servers that run the Kerberos Key Distribution Center (KDC) service. (For more information about the Kerberos v5 authentication protocol and the KDC, see "Authentication" in this book.)

Every Windows 2000-based domain controller also dynamically registers a single host resource record (an A resource record), which contains the name of the domain (*DnsDomainName*) where the domain controller is and the IP address of the domain controller. The A resource record makes it possible for clients that do not recognize SRV records to locate a domain controller by means of a generic host lookup.

You can disable the Net Logon registration of an A resource record that maps the Active Directory domain name to the IP address of the domain controller. For example, if a Web server registers the same name as the name of an Active Directory domain, you do not want non-Web servers to register A resource records for this name. Otherwise, if the Web browser located the domain controller instead of the Web server, the browser would receive the message that the site for which it was searching was not found. In another example, if a mail server is not enabled to do mail exchanger (MX) resource record lookup and, therefore, relies on A resource records for DNS lookup, the names that are used for mail servers must not be identical to the names that are used by other services, such as Active Directory.

To disable Net Logon registration of the A record for a domain controller

1. On the **Start** menu, click **Run**.
2. Type **regedt32.exe** or **regedit.exe**, and then click **OK**.
3. In the registry editor, navigate to HKEY_LOCAL_MACHINE \SYSTEM \CurrentControlSet \Services \Netlogon\Parameters.
4. If the entry exists, double-click the **DnsRegisterARecords** entry.

5. In the **DWORD Editor** dialog box (in Regedt32.exe) or the **Edit DWORD Value** (in Regedit.exe), type **0** in the text box, and then click **OK**.
6. If the entry does not exist, create the entry as follows:
   - In Regedt32.exe, on the **Edit** menu, click **Add Value**.

   In the Value Name box, type DnsRegisterARecords.

   In the Data Type drop-down list box, click REG_DWORD, and then click OK.

   In the DWORD Editor dialog box, type 0 in the Data box, and then click OK.

   - Or -

   - In Regedit.exe, on the **Edit** menu, click **New**.

   Click DWORD Value.

   Type DnsRegisterARecords for the value name. A value of 0 is assigned automatically.
7. Close the registry editor.

Caution Editing the registry directly can have serious, unexpected consequences that can prevent the system from starting and require that you reinstall Windows 2000. There are programs available in Control Panel or Microsoft Management Console (MMC) for performing most administrative tasks. These programs provide safeguards that prevent you from entering conflicting settings or settings that are likely to degrade performance or damage your system. Registry editors bypass the standard safeguards that are provided by these administrative tools. Modifying the registry is recommended only when no administrative tool is available. Before you make changes to the registry, it is recommended that you back up any valuable data on the computer. For instructions about how to edit registry entries, see Help for the registry editor that you are using. For more information about the registry, see the *Microsoft Windows 2000 Resource Kit* Technical Reference to the Windows 2000 Registry (Regentry.chm).

NetBIOS Domain Name Registration

A domain controller registers its NetBIOS name (*DomainName*[1C]) by broadcasting or directing a NetBIOS name registration request to a NetBIOS name server, such as a WINS server. Registering the NetBIOS name makes it possible for Windows-based clients that are not DNS-enabled to find the domain controllers that are running Windows 2000, Windows NT 4.0, or Windows NT 3.51. In this case, the client finds the domain controller by sending a Net Logon mailslot request that is based on the NetBIOS domain name.

Note NetBIOS recognizes domain controllers by the [1C] registration.

For more information about registering names with WINS, see "Windows Internet Name Service" in the *TCP/IP Core Networking Guide*.

SRV Resource Records

When a Windows 2000-based domain controller starts up, the Net Logon service uses dynamic updates to register SRV resource records in the DNS database, as described in "A DNS RR for specifying the location of services (DNS SRV)." For more information about this draft, see the Internet Engineering Task Force (IETF) link on the Web Resources page at http://windows.microsoft.com/windows2000/reskit/webresources . Follow the links to Internet Drafts, and then use a keyword search.

The SRV record is used to map the name of a service (in this case, the LDAP service) to the DNS computer name of a server that offers that service. In a Windows 2000 network, an LDAP resource record locates a domain controller.

A workstation that is logging on to a Windows 2000 domain queries DNS for SRV records in the general form:

_Service._Protocol.DnsDomainName

Active Directory servers offer the LDAP service over the TCP protocol; therefore, clients find an LDAP server by querying DNS for a record of the form:

_ldap._tcp.*DnsDomainName*

Note The service and protocol strings require an underscore (_) prefix to prevent potential collisions with existing names in the namespace.

_msdcs Subdomain

There are possible implementations of LDAP servers other than Windows 2000-based domain controllers. There are also possible implementations of LDAP directory services that employ Global Catalog servers but are not servers that are running Windows 2000. To facilitate locating Windows 2000-based domain controllers, in addition to the standard *_Service._Protocol.DnsDomainName* format, the Net Logon service registers SRV records that identify the well-known server-type pseudonyms "dc" (domain controller), "gc" (Global Catalog), "pdc" (primary domain controller), and "domains" (globally unique identifier, or GUID) as prefixes in the _msdcs subdomain. This Microsoft-specific subdomain allows location of domain controllers that have Windows 2000-specific roles in the domain or forest, as well as the location by GUID when a domain has been renamed. To accommodate locating domain controllers by server type or by GUID (abbreviated "dctype"), Windows 2000-based domain controllers register SRV records in the following form:

_Service._Protocol.DcType.**_msdcs.***DnsDomainName*

The addition of the _msdcs subdomain means that two sets of DNS names can be used to find an LDAP server: *DnsDomainName* is used to find an LDAP server or Kerberos server that is running TCP (or, in the case of a Kerberos server, either TCP or the User Datagram Protocol [UDP]), and the subdomain _msdcs.*DnsDomainName* is used to find an LDAP server that is running TCP and also functioning in a particular Windows 2000 role. The name "_msdcs" is reserved for locating domain controllers. The single keyword "_msdcs" was chosen to avoid cluttering the DNS namespace unnecessarily. Other constant, well-known names (pdc, dc, and gc) were kept short to avoid exceeding the maximum length of *DnsDomainNam*e.

SRV Records Registered by Net Logon

The list that follows provides the definitions of the names associated with registered SRV records. It also describes the lookup criteria supported by each record and the checks performed by Net Logon as each record is registered. Text in bold type denotes constant record components; text in italic type denotes variable names.

In the descriptions of registered SRV records, *DnsDomainName* refers to the DNS domain name that is used during creation of the domain controller when the domain tree is joined or created (that is, while the computer is running the Active Directory Installation Wizard). *DnsForestName* refers to the DNS domain name of the forest root domain.

The following is a list of the owner names of the SRV records that are registered by Net Logon. An owner name is the name of the DNS node to which the resource record pertains.

_ldap._tcp.*DnsDomainName***.**

Allows a client to locate a server that is running the LDAP service in the domain named by *DnsDomainName*. The server is not necessarily a domain controller - that is, the only assumption that can be made about the server is that it supports the LDAP application programming interface (API). All Windows 2000 Server-based domain controllers register this SRV record (for example, _ldap._tcp.reskit.com.).

_ldap._tcp.*SiteName***._sites.***DnsDomainName***.**

Allows a client to locate a server that is running the LDAP service in the domain named in *DnsDomainName* in the site named by *SiteName*. *SiteName* is the relative distinguished name of the site object that is stored in the Configuration container in Active Directory. The server is not necessarily a domain controller. All Windows 2000 Server-based domain controllers register this SRV record (for example, _ldap._tcp.charlotte._sites.reskit.com.).

_ldap._tcp.dc._msdcs.*DnsDomainName***.**

Allows a client to locate a domain controller (dc) of the domain named by *DnsDomainName*. All Windows 2000 Server-based domain controllers register this SRV record.

_ldap._tcp.*SiteName***._sites.dc._msdcs.***DnsDomainName***.**

Allows a client to locate a domain controller for the domain named by *DnsDomainName* and in the site named by *SiteName*. All Windows 2000 Server-based domain controllers register this SRV record.

_ldap._tcp.pdc._msdcs.*DnsDomainName***.**

Allows a client to locate the server that is acting as the primary domain controller (also known as a "PDC") in the mixed-mode domain named in *DnsDomainName*. Only the PDC emulator master of the domain (the Windows 2000-based domain controller that advertises itself as the primary domain controller to computers that need a primary domain controller) registers this SRV record.

_ldap._tcp.gc._msdcs.*DnsForestName***.**

Allows a client to locate a Global Catalog (gc) server for this forest. Only domain controllers that are functioning as Global Catalog servers for the forest named in *DnsForestName* register this SRV record (for example, _ldap._tcp.gc._msdcs.reskit.com.).

_ldap._tcp.*SiteName***._sites.gc._msdcs.***DnsForestName***.**

Allows a client to locate a Global Catalog (gc) server for this forest in the site named in *SiteName*. Only domain controllers that are serving as Global Catalog servers for the forest named in *DnsForestName* register this SRV record (for example, _ldap._tcp.charlotte._sites.gc._msdcs.reskit.com.).

_gc._tcp.*DnsForestName***.**

Allows a client to locate a Global Catalog (gc) server for this domain. The server is not necessarily a domain controller. Only a server that is running the LDAP service and functioning as the Global Catalog server for the forest named in *DnsForestName* registers this SRV record (for example, _gc._tcp.reskit.com.).

Note In Windows 2000, a Global Catalog server is a domain controller. Other non-Windows 2000 implementations of directory services can also register servers as Global Catalog servers.

_gc._tcp.*SiteName***._sites.***DnsForestName***.**

Allows a client to locate a Global Catalog (gc) server for this forest in the site named in *SiteName*. The server is not necessarily a domain controller. Only a server that is running the LDAP service and functioning as the Global Catalog server for the forest named in *DnsForestName* registers this SRV record (for example, _gc._tcp.charlotte._sites.reskit.com.).

_ldap._tcp.*DomainGuid***.domains._msdcs.***DnsForestName***.**

Allows a client to locate a domain controller in a domain on the basis of its GUID. A GUID is a 128-bit number that is automatically generated for referencing objects in Active Directory - in this case, the domain object. This operation is expected to be infrequent; it occurs only when the *DnsDomainName* of the domain has changed, the *DnsForestName* is known, and *DnsForestName* has not also been renamed (for example, _ldap._tcp.4f904480-7c78-11cf-b057-00aa006b4f8f.domains._msdcs.reskit.com.). All domain controllers register this SRV record.

_kerberos._tcp.*DnsDomainName***.**

Allows a client to locate a server that is running the Kerberos KDC service for the domain that is named in *DnsDomainName*. The server is not necessarily a domain controller. All Windows 2000 Server-based domain controllers that are running an RFC 1510-compliant Kerberos KDC service register this SRV record.

_kerberos._udp.*DnsDomainName***.**

Same as **_kerberos._tcp.***DnsDomainName*, except that UDP is implied.

_kerberos._tcp.*SiteName***._sites.***DnsDomainName***.**

Allows a client to locate a server that is running the Kerberos KDC service for the domain that is named in *DnsDomainName* and is also in the site named in *SiteName*. The server is not necessarily a domain controller. All Windows 2000 Server-based domain controllers that are running an RFC 1510-compliant Kerberos KDC service register this SRV record.

_kerberos._tcp.dc._msdcs.*DnsDomainName***.**

Allows a client to locate a domain controller that is running the Windows 2000 implementation of the Kerberos KDC service for the domain named in *DnsDomainName*. All Windows 2000 Server-based domain controllers that are running the KDC service (that is, that implement a public key extension to the Kerberos v5 protocol Authentication Service Exchange subprotocol) register this SRV record.

_kerberos.tcp.*SiteName***._sites.dc._msdcs.***DnsDomainName***.**

Allows a client to locate a domain controller that is running the Windows 2000 implementation of the Kerberos KDC service for the domain that is named in *DnsDomainName* and that is also in the site named in *SiteName*. All Windows 2000 Server-based domain controllers that are running the KDC service (that is, that implement a public key extension to the Kerberos protocol Authentication Service Exchange subprotocol) register this SRV record.

_kpasswd._tcp.*DnsDomainName***.**

Allows a client to locate a Kerberos Password Change server for the domain. All servers that provide the Kerberos Password Change service (which includes all Windows 2000-based domain controllers) register this name. This server at least conforms to "Kerberos Change Password Protocol." (For more information about this draft, see the Microsoft Platform SDK link on the Web Resources page at http://windows.microsoft.com/windows2000/reskit/webresources . Use a keyword search to locate the draft.) The server is not necessarily a domain controller. All Windows 2000 Server-based domain controllers that are running an RFC 1510-compliant Kerberos KDC service register this SRV record.

_kpasswd._udp.*DnsDomainName***.**

Same as **_kpasswd._tcp.***DnsDomainName*, except that UDP is implied.

If multiple domain controllers have the same criteria, multiple records exist with the same owner name. A client that is looking for a domain controller with specific criteria would receive all the applicable records from the DNS server. The client would pick one of the returned records to select a domain controller, as described in "A DNS RR for specifying the location of services (DNS SRV)." For more information about this draft, see the Internet Engineering Task Force (IETF) link on the Web Resources page at http://windows.microsoft.com/windows2000/reskit/webresources . Follow the links to Internet Drafts, and then use a keyword search.

For information about the Kerberos v5 authentication protocol and Kerberos subprotocol extensions, see "Authentication" in this book.

Host Records for Non-SRV-Aware Clients

Net Logon registers the following DNS A records for the use of LDAP clients that do not support DNS SRV records (that is, that are "non-SRV-aware"). The Locator does not use these records.

The following owner names of A (host) records are registered by Net Logon:

*DnsDomainName***.**

Allows a non-SRV-aware client to locate any domain controller in the domain by looking up an A record. A name in this form is returned to the LDAP client through an LDAP referral. (For more information about LDAP referrals, see "LDAP Referrals" later in this chapter.) A non-SRV-aware client looks up the name; an SRV-aware client looks up the appropriate SRV resource record.

gc._msdcs.*DnsForestName***.**

Allows a non-SRV-aware client to locate any Global Catalog server in the forest by looking up an A record. A name in this form is returned to the LDAP client through an LDAP referral. A non-SRV-aware client looks up this name; an SRV-aware client looks up the appropriate SRV resource record.

Net Logon also registers a DNS CNAME (alias) record for use by Active Directory replication. The Locator does not use this record.

The owner name of the CNAME record is:

*DsaGuid***._msdcs.***DnsForestName***.**

Allows a client to locate any domain controller in the forest by looking up an A record. The only information that is known about the domain controller is the GUID of the directory system agent (also known as the "DSA") object for the domain controller and the name of the forest in which the domain controller is located. This record is used to facilitate renaming a domain controller.

Other SRV Record Content

The following information is also included in an SRV record:

Priority The priority of the server. Clients attempt to contact the server with the lowest priority.

Weight A load-balancing mechanism that is used when selecting a target host from those that have the same priority. Clients randomly choose SRV records that specify target hosts to be contacted, with probability proportional to the weight

Port Number The port where the server is listening for this service.

Target The fully qualified domain name of the host computer.

The following example illustrates the combined information that is contained in A resource records and SRV resource records. A domain controller named Phoenix in the domain reskit.com has an IP address of 157.55.81.157. It registers the following A records and SRV records with DNS:

```
phoenix.reskit.com A 157.55.81.157
_ldap._tcp.reskit.com SRV 0 0 389 phoenix.reskit.com
_kerberos._tcp.reskit.com SRV 0 0 88 phoenix.reskit.com
_ldap._tcp.dc._msdcs.reskit.com SRV 0 0 389 phoenix.reskit.com
_kerberos._tcp.dc._msdcs.reskit.com SRV 0 0 88 phoenix.reskit.com.
```

When the appropriate SRV records and A records are in place, a DNS lookup of _ldap._tcp.dc._msdcs.reskit.com returns the names and addresses of all domain controllers in the domain.

For more information about A records, SRV records, DNS, and dynamic updates, see "Introduction to DNS" and "Windows 2000 DNS" in the *TCP/IP Core Networking Guide*.

Domain Controller Location Process

Each Windows 2000-based domain controller registers its DNS domain name on the DNS server and registers its NetBIOS name by using a transport-specific mechanism (for example, WINS). Therefore, a DNS client locates a domain controller by querying DNS, and a NetBIOS client locates a domain controller by querying the appropriate transport-specific name service. Because the code for the Windows 2000 IP/DNS-compatible Locator and the Windows NT 4.0-compatible Locator is shared, both DNS clients and NetBIOS clients are supported.

The process for locating a domain controller can be summarized as follows:

1. On the client (the computer that is locating the domain controller), the Locator is initiated as a remote procedure call (RPC) to the local Net Logon service. The Locator API (DsGetDcName) is implemented by the Net Logon service.
2. The client collects the information that is needed to select a domain controller and passes the information to the Net Logon service by using the DsGetDcName API.
3. The Net Logon service on the client uses the collected information to look up a domain controller for the specified domain in one of two ways:
   - For a DNS name, Net Logon queries DNS by using the IP/DNS-compatible Locator - that is, DsGetDcName calls DnsQuery to read the SRV records and A records from DNS after it appends an appropriate string to the front of the domain name that specifies the SRV record.
   - For a NetBIOS name, Net Logon performs domain controller discovery by using the Windows NT 4.0-compatible Locator - that is, by using the transport-specific mechanism (for example, WINS).

   Note In Windows NT 4.0 and earlier, "discovery" is a process for locating a domain controller for authentication in either the primary domain or a trusted domain.
4. The Net Logon service sends a datagram to the discovered domain controllers ("pings" the computers) that register the name. For NetBIOS domain names, the datagram is implemented as a mailslot message. For DNS domain names, the datagram is implemented as an LDAP UDP search.
5. Each available domain controller responds to the datagram to indicate that it is currently operational and then returns the information to DsGetDcName.
6. The Net Logon service returns the information to the client from the domain controller that responds first.
7. The Net Logon service caches the domain controller information so that it is not necessary to repeat the discovery process for subsequent requests. Caching this information encourages the consistent use of the same domain controller and, thus, a consistent view of Active Directory.

DsGetDcName API

The following parameters are the DsGetDcName API parameters that Net Logon uses to collect information from the client and to compose the DNS or WINS query. As described in "Domain Controller Location Process" earlier in this chapter, the API is called remotely in a datagram that is sent to the discovered domain controllers, and the domain controller provides the information to the client.

ComputerName The name of the computer that collects the information. The value of this parameter is usually NULL, which denotes the local computer. The DsGetDcName API is passed by RPC to the specified computer.

DomainName The name of the domain to be queried. This name can be either a DNS-style name (for example, reskit.com.) or a flat, NetBIOS-style name (for example, reskit). If a DNS-style name is specified, the name can be specified with or without a trailing dot.

DomainGuid The GUID of the domain being queried. This value is used when a domain has been renamed. If this value is specified and *DomainName* has been renamed, DsGetDcName attempts to locate a domain controller in the domain that has the specified *DomainGuid*.

SiteName The name of the site in which the domain controller that is returned should be located. This parameter is usually not specified. When the site is not specified, the domain controller that is returned is in the site that is closest to the one in which *ComputerName* is located.

Flags Additional information that the application can use to process the request. Flags include the following:

DS_FORCE_REDISCOVERY. Requires that a domain controller be determined, even if a domain controller is currently known in the cache. This flag can be used when an additional domain controller becomes available or when an existing domain controller has been detected to be unavailable. This function guarantees only the domain controller that was returned when the domain controller was initially entered into the cache. The DS_FORCE_REDISCOVERY flag should not be specified unless this function has been called recently without the flag. An attempt to gain access to the cached domain controller should be made. Only if this initial attempt to gain access fails should the flag be used to call the function again.

DS_DIRECTORY_SERVICE_REQUIRED. Requires that the returned domain controller support Directory Server API (is running Windows 2000 Server).

DS_DIRECTORY_SERVICE_PREFERRED. Prefers that the returned domain controller support Active Directory (is running Windows 2000 Server). If no such domain controller is available, a domain controller that is running Windows NT 4.0 or earlier is returned. If no domain controller that supports a directory service is available, DsGetDcName returns the name of the closest non-Active Directory domain controller; however, DsGetDcName returns the non-Active Directory domain controller information only after the attempt to find an Active Directory domain controller has timed out.

DS_GC_SERVER_REQUIRED. Requires that the returned domain controller be a Global Catalog server for the forest of domains that has the specified domain as the root. This flag cannot be set if the DS_PDC_REQUIRED flag is set.

DS_PDC_REQUIRED. Requires that the returned domain controller be the primary domain controller for the domain. This flag cannot be set if the DS_GC_SERVER_REQUIRED flag is set. If this flag is specified, the DS_DIRECTORY_SERVICE_PREFERRED flag and DS_WRITABLE_REQUIRED flag are ignored.

DS_WRITABLE_REQUIRED. Requires that the returned domain controller host a writable copy of Active Directory (or Security Accounts Manager [SAM]). If the specified *DomainName* is a NetBIOS name, this flag causes DsGetDcName to find either a primary domain controller or a Windows 2000 Server-based domain controller. If the specified *DomainName* is a DNS name, this flag is ignored.

DS_IP_REQUIRED. Requires that the IP address of the discovered domain controller be returned.

DS_KDC_REQUIRED. Requires that the returned domain controller currently be running the KDC service.

DS_TIMESERV_REQUIRED. Requires that the returned domain controller be currently running the Windows Time Service.

DS_GOOD_TIMESERV_PREFERRED. Prefers that the returned domain controller be a "reliable" time server. The Windows Time Service can be configured to declare one or more domain controllers as "reliable" time servers. This flag is intended for use only by the Windows Time Service. The behavior of the flag is subject to change to achieve the implementation that best supports the Windows Time Service (W32time).

DS_IS_FLAT_NAME. Specifies that the *DomainName* parameter be a NetBIOS name. As such, the IP/DNS-compatible Locator is not tried, and the Windows NT 4.0-compatible Locator is used. (For more information about locating a domain controller by using a NetBIOS name, see "Windows NT 4.0-Compatible Locator Process for Non-IP/DNS Clients" later in this chapter.) This flag cannot be specified with the DS_IS_DNS_NAME flag. It is valid to set neither DS_IS_FLAT_NAME nor DS_IS_DNS_NAME; however, DsGetDcName () takes longer to find a domain controller because it must try both the DNS-style and NetBIOS names. In addition, it is potentially ambiguous to specify neither flag. For example, if you specify a domain name of "reskit," a domain with a NetBIOS name of "reskit" exists on your network, and a different domain with a DNS-style name of "reskit" also exists on your network, DsGetDcName() might find a domain controller in either domain.

DS_IS_DNS_NAME. Specifies that the *DomainName* parameter is a DNS name. (For more information about the *DomainName* parameter, see "Windows NT 4.0-Compatible Locator Process for Non-IP/DNS Clients" later in this chapter.) This flag cannot be specified by using the DS_IS_FLAT_NAME flag.

DS_AVOID_SELF. Specifies that the domain controller returned should not be the domain controller named by *ComputerName*. If *ComputerName* is not a domain controller, this flag is ignored. This flag can be used to get the name of another domain controller in the domain.

DS_RETURN_FLAT_NAME. Specifies that the returned *DomainControllerName* and returned *DomainName* must be flat names. If a flat name is not available, an error message is returned. This flag cannot be specified by using the DS_RETURN_DNS_NAME flag.

DS_RETURN_DNS_NAME. Specifies that the returned *DomainControllerName* and returned *DomainName* must be DNS names. If a DNS name is not available, an error message is returned. This flag cannot be specified with the DS_RETURN_FLAT_NAME flag. This flag implies the DS_IP_REQUIRED flag.

DS_ONLY_LDAP_NEEDED. Specifies that the server returned must be an LDAP server. The server returned is not necessarily a domain controller. No other services are implied to be present at the server. The server returned does not necessarily have either a writable Configuration container or a writable Schema container. The server returned is not necessarily used to create or modify security principles. This flag can be used with the DS_GC_SERVER_REQUIRED flag to return an LDAP server that also hosts a Global Catalog server. The returned Global Catalog server is not necessarily a domain controller. No other services are implied to be present at the server. If this flag is specified, the following flags are ignored: DS_PDC_REQUIRED, DS_TIMESERV_REQUIRED, DS_GOOD_TIMESERV_PREFERRED, and DS_KDC_REQUIRED.

For more information about the DsGetDcName API and DsGetDcName flags, see the Microsoft Platform SDK link on the Web Resources page at http://windows.microsoft.com/windows2000/reskit/webresources .

Finding a Domain Controller in the Closest Site

During a search for a domain controller, the Locator attempts to find a domain controller in the site closest to the client. When the domain that is being sought is a Windows 2000 domain, the domain controller uses the information stored in Active Directory to determine the closest site. When the domain being sought is a Windows NT 4.0 domain, domain controller discovery occurs when the client starts and uses the first domain controller that it finds.

As described in "SRV Records Registered by Net Logon" earlier in this chapter, each Windows 2000-based domain controller registers DNS records that indicate the site where the domain controller is located. The site name (the relative distinguished name of the site object in Active Directory) is registered in several records so that the various roles the domain controller might perform (for example, Global Catalog server or Kerberos server) can be associated with the domain controller's site. When DNS is used, the Locator searches first for a site-specific DNS record before it begins to search for a DNS record that is not site-specific (thereby preferentially locating a

domain controller in that site).

A client computer stores its own site information in the registry, but the computer is not necessarily located physically in the site associated with its IP address. For example, a portable computer that was moved to a new location contacts a domain controller in its home site, which is not the site to which the computer is currently connected. In this situation, the domain controller looks up the client site on the basis of the client IP address by comparing the address to the sites that are identified in Active Directory, and returns the name of the site that is closest to the client. The client then updates the information in the registry.

The domain controller stores site information for the entire forest in the Configuration container. The domain controller uses the site information to check the IP address of the client computer against the list of subnets in the forest. In this way, the domain controller ascertains the name of the site in which the client is assumed to be located, or the site that is the closest match, and returns this information to the client.

Active Directory Site and Subnet Objects

A site is a collection of subnets that have high-speed connections. In Active Directory, a site is defined by a site object in the cn=Sites,cn=Configuration,dc=*ForestRootDomain* container. A subnet is an addressed segment within a site and is represented by an object in the cn=Subnets,cn=Sites,cn=Configuration,dc=*ForestRootDomain* container.

The site in which a domain controller is located is identified in the Configuration container by the domain controller object that is located within the cn=Servers container beneath the site object for a particular site. A domain controller can identify the site of a client by using the subnet object in the Sites container. Each subnet object has a *siteObject* property ("attribute") that links it to a site object; the value of the *siteObject* property is the distinguished name of the site object. This link enables a domain controller to identify clients that have an IP address in the specified subnet as being in the specified site.

Subnet names in Active Directory take the form "network/bits masked" (for example, the subnet object 172.16.72.0/22 has a subnet of 172.16.72.0 and a 22-bit subnet mask). If this subnet had a *siteObject* property value that contained the distinguished name of the Seattle site object, all IP addresses in the 172.16.72.0/22 subnet would be considered to be in the Seattle site. The *siteObject* property is a single value, which implies that a single subnet maps to a single site. However, multiple subnet objects can be linked to the same site object. The directory administrator manually creates subnet objects and, hence, the *siteObject* property value.

The Configuration container (including all of the site and subnet objects in it) is replicated to all domain controllers in the forest. Therefore, any domain controller in the forest can identify the site in which a client is located, compare it to the site in which the domain controller is located, and indicate to the client whether that domain controller's site is the closest site to the client.

For more information about site and subnet objects, see "Active Directory Replication" in this book. For more information about networks, subnets, and subnet masks, see "Introduction to TCP/IP" in the *TCP/IP Core Networking Guide*.

Mapping IP Addresses to Site Names

During Net Logon startup, the Net Logon service on each domain controller enumerates the site objects in the Configuration container. Net Logon on each domain controller is also notified of any changes made to the site objects. Net Logon uses the site information to build an in-memory structure that is used to map IP addresses to site names.

When a client that is searching for a domain controller receives the list of domain controller IP addresses from DNS, the client begins querying the domain controllers in turn to find out which domain controller is available and appropriate. Active Directory intercepts the query, which contains the IP address of the client, and passes it to Net Logon on the domain controller. Net Logon looks up the client IP address in its subnet-to-site mapping table by finding the subnet object that most closely matches the client IP address and then returns the following information:

- The name of the site in which the client is located, or the site that most closely matches the client IP address.
- The name of the site in which the current domain controller is located.
- A bit that indicates whether the found domain controller is located (bit is set) or not located (bit is not set) in the site closest to the client.

The domain controller returns the information to the client. The response also contains various other pieces of information that describe the domain controller. The client inspects the information to determine whether to try to find a better domain controller. The decision is made as follows:

- If the returned domain controller is in the closest site (the returned bit is set), the client uses this domain controller.
- If the client has already tried to find a domain controller in the site in which the domain controller claims the client is located, the client uses this domain controller.
- If the domain controller is not in the closest site, the client updates its site information and sends a new DNS query to find a new domain controller in the site. If the second query is successful, the new domain controller is used. If the second query fails, the original domain controller is used.

If the domain that is being queried by a computer is the same as the domain to which the computer is joined, the site in which the computer resides (as reported by a domain controller) is stored in the computer registry. The client stores this site name in the **DynamicSiteName** registry entry in HKEY_LOCAL_MACHINE \SYSTEM \CurrentControlSet \Services

Netlogon\Parameters. Therefore, the DsGetSiteName API returns the site in which the computer is located.

Never change dynamically determined values. To override the dynamic site name, add the **SiteName** entry with the REG_SZ data type in HKEY_LOCAL_MACHINE \SYSTEM \CurrentControlSet \Services \Netlogon \Parameters. When a value is present for the **SiteName** entry, the **DynamicSiteName** entry is not used. For more information about **SiteName** and **DynamicSiteName**, see the *Microsoft Windows 2000 Resource Kit* Technical Reference to the Windows 2000 Registry (Regentry.chm).

Caution Editing the registry directly can have serious, unexpected consequences that can prevent the system from starting and require that you reinstall Windows 2000. There are programs available in Control Panel or Microsoft Management Console (MMC) for performing most administrative tasks. These programs provide safeguards that prevent you from entering conflicting settings or settings that are likely to degrade performance or damage your system. Registry editors bypass the standard safeguards that are provided by these administrative tools. Modifying the registry is recommended only when no administrative tool is available. Before you make changes to the registry, it is recommended that you back up any valuable data on the computer. For instructions about how to edit registry entries, see Help for the registry editor that you are using. For more information about the registry, see the *Microsoft Windows 2000 Resource Kit* Technical Reference to the Windows 2000 Registry (Regentry.chm).

If the domain being located is the same as the domain to which the computer is joined and the computer has not physically moved to a different site since the last query, the dynamically determined site name in the registry is the actual site in which the computer is located. As such, the client finds a domain controller in the correct site without having to retry the operation. If the site name in the registry is not the current site of the computer (for example, if the computer is portable), the domain controller location process serves to update the site information in the registry.

Automatic Site Coverage

There is not necessarily a domain controller in every site. For various reasons, it is possible that no domain controller exists for a particular domain at the local site. By default, each domain controller checks all sites in the forest and then checks the replication cost

matrix. A domain controller advertises itself (registers a site-related SRV record in DNS) in any site that does not have a domain controller for that domain and for which its site has the lowest-cost connections. This process ensures that every site has a domain controller that is defined by default for every domain in the forest, even if a site does not contain a domain controller for that domain. The domain controllers that are published in DNS are those from the closest site (as defined by the replication topology).

For example, given one domain and three sites, a domain controller for that domain might be located in two of the sites, but there might be no domain controller for the domain in the third site. Replication to the domain that does not have a domain controller in the third site might be too expensive in terms of cost or replication latency. To ensure that a domain controller can be located in the site closest to a client computer, if not the same site, Windows 2000 automatically attempts to register a domain controller in every site. The algorithm that is used to accomplish automatic site coverage determines how one site can "cover" another site when no domain controller exists in the second site.

Determining Site Coverage on the Basis of Cost

Given one domain and sites A, B, and C, site A has no domain controllers for the domain. If a client in site A attempts to locate a domain controller, which domain controller should be returned? The answer depends on which site covers site A for the domain. Site coverage is determined according to site-link costs, and domain controllers register themselves in sites accordingly.

In the example, a site link exists between site A and both of the other sites - that is, the connections between domain controllers in site A, site B, and site C are configured for replication over site links in Active Directory Sites and Services. (For more information about site links and site-link costs, see "Active Directory Replication" in this book.) Costs are associated with site links based on the expense of transferring data over the connections. The administrator uses the speed of the connection between sites to assign a cost to the communication link, and replication uses the cost to establish the least expensive route for replication traffic.

Site A and site B are connected by site link AB. Site A and site C are connected by site link AC, with the following costs:

- Site link AB cost = 50.
- Site link AC cost = 100.

The link between site A and site C has a much higher cost than the link between site A and site C. The administrator configured this cost based on the expensive Integrated Services Digital Network (ISDN) line that connects site A and site C, and the administrator would prefer that resources in site B be used when possible. The site coverage algorithm ensures that a domain controller in site B registers itself as a domain controller for site A. In this way, clients in Site A that are looking for a domain controller find one from site B, instead of possibly finding one from site C. For more information about site link cost, see "Active Directory Replication" in this book.

Site Coverage Algorithm

During registration of SRV records in DNS, the following algorithm is used to determine which domain controllers register site SRV records that designate them as preferred domain controllers in sites that do not have a specific domain represented.

For every domain controller in the forest, follow this procedure:

1. Build a list of *target sites* - sites that have no domain controllers for this domain (the domain of the current domain controller).
2. Build a list of *candidate sites* - sites that have domain controllers for this domain.
3. For every target site, follow these steps:
    - Build a list of candidate sites of which this domain is a member. (If none, do nothing.)
    - Of these, build a list of sites that have the lowest site link cost to the target site. (If none, do nothing.)
    - If more than one, break ties (reduce this list to one candidate site) by choosing the site with the largest number of domain controllers.
    - If more than one, break ties by choosing the site that is first alphabetically.
    - Register target-site-specific SRV records for the domain controllers for this domain in the selected site.

Cache Time-out and Closest Site

If a domain member computer requests a domain controller while all domain controllers in its site are offline, the Locator necessarily returns a domain controller in a different site. The location of this domain controller is stored in the client cache. The cache lifetime is controlled by the **CloseSiteTimeout** entry in the registry.

In addition, the domain controller performs authentication, and a secure channel is set up. On subsequent location attempts, the lifetime of the cache and the lifetime of the secure channel are secondary to the location of a domain controller in the closest site.

If the domain controller that is stored in the client cache is not in a site that is close to the client, Net Logon attempts to find a close domain controller when either of the following events occurs:

- An interactive logon process uses pass-through authentication on the secure channel.
- The value in the **CloseSiteTimeout** registry entry has elapsed since the last attempt, and any other attempt is made to use the secure channel (for example, pass-through authentication of network logons).

Thus, Net Logon attempts to find a close domain controller only on demand. The default value of the **CloseSiteTimeout** period is 15 minutes; the maximum value is 49 days, and the minimum value is 60 seconds. The implications of this setting are that if the time-out value is too large, a client never tries to find a close domain controller if there is not one available at startup. If the value of this setting is too small, secure channel traffic is unnecessarily slowed down by discovery attempts.

For more information about creating the **CloseSiteTimeout** entry, see the *Microsoft Windows 2000 Resource Kit* Technical Reference to the Windows 2000 Registry (Regentry.chm).

Clients with No Apparent Site

Sometimes the client pings a domain controller and the client IP address cannot be found in the subnet-to-site mapping table. In this case, the domain controller returns a NULL site name, and the client uses the returned domain controller.

For more information about locating sites, see "Active Directory Replication" in this book.

Types of Locators

On the basis of parameters passed to Net Logon in the DsGetDcName API, the process of locating a domain controller proceeds in one of two ways:

- The IP/DNS-compatible Locator is used if the domain name passed to DsGetDcName is a DNS-compatible name. The Net Logon service on the client looks up the name in DNS (by calling DnsQuery) after it appends an appropriate string to the front of the domain name. The DNS service supports a query for determining the set of domain controllers. If the client site name is known, the client DNS query specifies the site. DNS returns the IP addresses of domain controllers that match the DNS query. The client Net Logon service sends an LDAP UDP message to one or more of the domain controllers that have been returned by DNS in order to determine whether any of the specified domain controllers are running and support the specified domain.
- The Windows NT 4.0-compatible Locator is used if the domain name passed to DsGetDcName is a NetBIOS name. The Net Logon

service on the client sends a transport-specific logon request query to locate a domain controller in a particular domain and then sends a mailslot message to one or more of the domain controllers to determine whether any of the domain controllers it found are running and support the specified domain.

For more information about the DsGetDcName Locator API, see the Microsoft Platform SDK link on the Web Resources page at http://windows.microsoft.com/windows2000/reskit/webresources. For more information about DNS and IP address resolution, see "Introduction to DNS" and "Windows 2000 DNS" in the *TCP/IP Core Networking Guide*.

IP/DNS-Compatible Locator Process for Windows 2000 Clients

The IP/DNS-compatible Locator uses DNS SRV records to locate the closest domain controller on the basis of the site name registered for a particular domain controller.

Note A site name is registered in DNS as a DNS label, which implies that the site name must syntactically conform to the rules of a DNS label. For example, a label cannot be more than 63 octets long and cannot contain illegal characters such as a dot. (For more information about DNS naming rules, see "Introduction to DNS" and "Windows 2000 DNS" in the *TCP/IP Core Networking Guide*.)

The IP/DNS-compatible Locator algorithm runs in the context of the Net Logon service that is running (typically) on the client. The following process takes place after the Net Logon service has established that a DNS name is being requested.

1. Locator queries DNS (calls DnsQuery) and specifies one of the criteria-specific DNS host names. If the client has site information, it first queries the site-specific DNS names. On the basis of parameters to the DsGetDcName API and the success or failure of a previous lookup, DNS is queried as follows:

   Note When DsGetDcName is called without specifying a site, DsGetDcName always attempts to find a domain controller in the site of the client or the site closest to the client, as described in "Automatic Site Coverage" earlier in this chapter. The details of site-specific discovery are omitted in the discussion that follows.

   - If the DS_PDC_REQUIRED flag is specified, look up the **_ldap._tcp.pdc._msdcs.***DnsDomainName* name. Return any success or failure to the caller.
   - If the DS_GC_SERVER_REQUIRED flag is specified and the *SiteName* parameter is specified, look up the **_ldap._tcp.***SiteName*.**_sites.gc._msdcs.***DnsForestName* name. If no domain controller is found, go to the next step. Otherwise, return any success or failure to the caller.
   - If the DS_GC_SERVER_REQUIRED flag is specified and the *SiteName* parameter is not specified, look up the **_ldap._tcp.gc._msdcs.***DnsForestName* name. Return any success or failure to the caller.
   - If the DS_KDC_REQUIRED flag is specified and the *SiteName* parameter is specified, look up the **_kerberos._tcp.***SiteName***._sites.dc._msdcs.***DnsForestName* name. If no domain controller can be found, go to the next step. Otherwise, return any success or failure to the caller.
   - If the DS_KDC_REQUIRED flag is specified and the *SiteName* parameter is not specified, look up the **_kerberos._tcp.dc._msdcs.***DnsForestName* name. Return any success or failure to the caller.
   - If the DS_ONLY_LDAP_NEEDED flag is specified and the *SiteName* parameter is specified, look up the **_ldap._tcp.***SiteName***._sites.***DnsDomainName* name. If no computer is found, go to the next step. Otherwise, return any success or failure to the caller.
   - If the DS_ONLY_LDAP_NEEDED flag is specified and the *SiteName* parameter is not specified, look up the **_ldap._tcp.***DnsDomainName* name. Return any success or failure to the caller.
   - If the *SiteName* parameter is specified, look up the **_ldap._tcp.***SiteName***._sites.dc._msdcs.***DnsDomainName* name. If no domain controller is found, go to the next step. Otherwise, return any success or failure to the caller.
   - If the *SiteName* parameter is not specified, look up the **_ldap._tcp.dc._msdcs.***DnsDomainName* name. If no domain controller has that name (which is not the same as "If no domain controllers can be found"), go to the next step. Otherwise, return any success or failure to the caller.
   - If the DomainGuid parameter is specified, look up the **_ldap._tcp.***DomainGuid***.domains._msdcs.***DnsForestName* name. Otherwise, return any success or failure to the caller.
2. If IP is not supported or DNS is not supported (indicated by an error message that is returned from the DnsQuery API), call the Windows NT 4.0-compatible Locator.
3. If the specified name cannot be found (perhaps because the domain has been renamed), call the Windows NT 4.0-compatible Locator.
4. DNS returns a list of IP addresses that match the target domain in the SRV records (that is, IP addresses of domain controllers in the specified domain) that are sorted by priority and weight, as described the Internet Draft "A DNS RR for specifying the location of services (DNS SRV)." (For more information about this draft, see the Internet Engineering Task Force [IETF] link on the Web Resources page at http://windows.microsoft.com/windows2000/reskit/webresources. Follow the links to Internet Drafts, and then use a keyword search.) The client pings each IP address in the order returned. The ping is a UDP LDAP query to port 389. The client pings each domain controller from the list. After each ping, the client waits one-tenth of a second for a response to the ping (or to any previous ping), and then pings the next domain controller. Choosing the domain controllers at random provides a first level of load balancing. Doing multiple pings in quick succession ensures that the discovery algorithm terminates in a finite amount of time.
5. The pinging continues until a viable response has been received or all of the returned domain controllers have been tried.
6. When a domain controller responds to the ping, the information supplied in the response is compared to the information specified to DsGetDcName. If the information does not match, the response is ignored.
7. The first domain controller to respond to a ping is returned to the caller.

Windows NT 4.0-Compatible Locator Process for Non-IP/DNS Clients

When the IP/DNS Locator cannot be used, domain location proceeds by using the Windows NT 4.0-compatible Locator. The IP/DNS Locator is not used under any of the following conditions:

- A NetBIOS domain name is supplied to DsGetDcName.
- IPX or NetBEUI is the only available transport.
- A Windows NT 3.51 or Windows NT 4.0 domain controller is being located.
- A Windows 3.1, Windows for Workgroups 3.1, Windows for Workgroups 3.11, Windows NT 3.51, Windows NT 4.0, Windows 95, or Windows 98 client is doing the locating.

In these cases, the Windows NT 4.0-compatible Locator is used to locate a domain controller in the domain. In general, the Windows NT 4.0-compatible Locator works as follows:

1. The client discovers a domain controller in a domain by sending a NETLOGON_SAM_LOGON_REQUEST message to \mailslot\net\ntlogon mailslot on the *DomainName*[1C] NetBIOS name of the domain whose domain controller is being discovered.
2. The *DomainName*[1C] NetBIOS name is registered by every Windows NT domain controller in the domain. The specific name

resolution mechanism and datagram delivery mechanism that are used are transport-specific. The client uses the first domain controller that responds to the message.

Note The algorithm described here locates any domain controller in the domain. A similar algorithm, implemented in NetGetDcName in Windows NT 4.0, is used when the caller specifies finding a primary domain controller.

Domain Controller Request

A client requests a domain controller by using the NETLOGON_SAM_LOGON_REQUEST message. In Windows 2000, this message is enhanced to contain the following new information:

- The **NtVersion** field specifies that Windows 2000-specific fields be present and that the response must contain more information.
- The **RequestedDomainName** field is used by Net Logon to identify the queried domain if the **RequestedDomainGuid** field is not present. This field immediately follows the existing **DomainSid** field.
- The **RequestedDomainGuid** field is used by Net Logon to identify the queried domain. This field immediately follows the **RequestedDomainName** field.

A Windows NT 3.51-based or Windows NT 4.0-based domain controller ignores the additional information in the query message.

Note When a Windows 2000 domain has an external trust relationship with a non-Windows 2000 domain (a Windows NT 3.51 or Windows NT 4.0 domain), discovery of domain controllers in the external trusted domain is performed immediately when the client starts. Discovery at logon is not possible because the accounts in the trusted domain, which is outside the forest, are not available in the Global Catalog. The Global Catalog is required for logging on to the domain.

Domain Controller Response

A domain controller responds to the NETLOGON_SAM_LOGON_REQUEST message with a NETLOGON_SAM_LOGON_RESPONSE mailslot message as follows:

1. A Windows NT 3.51-based or Windows NT 4.0-based domain controller returns a NETLOGON_SAM_LOGON_RESPONSE mailslot message.
2. A Windows 2000-based domain controller returns a NETLOGON_SAM_LOGON_RESPONSE mailslot message to a client that is running Microsoft® Windows® version 3.1, Microsoft® Windows® for Workgroups version 3.1, Microsoft® Windows® for Workgroups version 3.11, Windows NT 3.51, or Windows NT 4.0.
3. A Windows 2000-based domain controller returns a NETLOGON_SAM_LOGON_RESPONSE_EX message to a Windows 2000-based client. All characters in this message are in the UTF-8 character set. The response message contains the following additional information:

- The **NtVersion** field indicates that Windows 2000-specific fields are present.
- The **DomainGuid** field returns the GUID of the domain.
- The **DnsDomainName** field returns the DNS name of the domain.
- The **DnsForestName** field returns the DNS domain name of the forest in which the domain controller is located.
- The **DcSiteName** field returns the name of the site in which the domain controller is located.
- The **ClientSiteName** field returns the name of the site in which the client is located.
- The **DcSockAddr** field returns the IP address of the domain controller.
- The **Flags** field indicates the following information about the domain controller:

  Whether it is the primary domain controller.

  Whether it supports the Global Catalog.

  Whether it supports Active Directory.

  Whether it is in the site closest to the client.

The Windows NT 4.0-compatible Locator cannot completely implement the DsGetDcName flag, which requires that a domain controller that supports the directory service API is returned (that is, the returned domain controller must be running Windows 2000). When the DomainGuid parameter is NULL, WINS is configured, and no Windows 2000-based domain controllers are among the 25 closest domain controllers in the domain, then DsGetDcName fails as though no domain controller were available.

Requerying to Find the Closest Site

A client that is running Windows 2000 and using the Windows NT 4.0-compatible Locator to find a domain controller attempts to find a domain controller in the site closest to it. After the client finds a domain controller, the mailslot response is inspected to determine the following:

- Whether the DNS domain name of the domain was returned. (The value is not TRUE if the domain controller is running Windows NT 3.51 or Windows NT 4.0.)
- Whether the ClientSiteName was returned. (The value is not TRUE if there is no subnet object that matches the IP address of the client.)
- Whether the domain controller did not indicate that it is in the site closest to the client.

If all of these conditions have the value TRUE, DsGetDcName calls DsGetDcName again and passes the DNS domain name and the site name. This new call to DsGetDcName uses the IP/DNS Locator to try to find a domain controller in the named site. If the new call finds a domain controller, that domain controller is returned to the client as the result of the original DsGetDcName call. If the new call does not find a domain controller, the original domain controller is returned.

Finding Information in Active Directory

Client applications use various mechanisms to find information in Active Directory. Most requests for directory objects are carried out either through the Active Directory Service Interfaces (ADSI) LDAP provider or through the LDAP API. Ultimately, every request is subject to the LDAP rules for locating objects. Active Directory processes LDAP requests for locally stored directory information (that is, information specific to the current domain directory partition) and implements a referral mechanism to locate objects stored in other directory partitions. If the object does not exist, an error is returned that states that the object is not in the directory.

Resolving Names in Directory Operations

When any directory operation is requested by a client, the domain controller that is contacted resolves names by using its "knowledge" of the entire directory to determine whether the domain controller can complete the operation or whether it must refer the client to another server for part or all of the operation.

LDAP finds an object in the directory according to the path that is specified in the distinguished name (also known as the "DN") of the object. Every object is stored in the directory database according to its relative distinguished name (also known as the "RDN") and parent identifier, not according to its distinguished name. A distinguished name is a series of relative distinguished names that lead

from the object's relative distinguished name to the relative distinguished name at the top of the distinguished name hierarchy. Therefore, if you know the relative distinguished name of an object, you can always determine the full distinguished name by following the references to the parent objects and ultimately to the root object. For example, the distinguished name of a user object might be cn=*UserName*,ou=*OrganizationalUnit*,dc=*DomainName*,...dc=*DomainName*, where the series of relative distinguished names denoted by dc=*DomainName* identifies the DNS domain of the object. This portion of the distinguished name can be matched to the tree of domain names that is formed by certain attribute values that are stored in cn=Partitions,cn=Configuration, dc=*ForestRootDomain*.

Note The objects in cn=Partitions,cn=Configuration,dc=*ForestRootDomain* are cross-reference objects; they contain information that Active Directory can use to construct the directory tree hierarchy.

Because every domain controller has the information about all directory partitions in the forest, splitting a distinguished name into a suffix (which identifies the relative path within the domain) and a prefix (the dc= components that identify the domain itself) is always a local operation. If the local domain controller stores a copy of the domain in question, the domain controller can verify the prefix of the distinguished name and perform the requested operation. If the local domain controller does not store a copy of the domain in question, it returns either a referral to another server or an error message that states that the object does not exist.

Components of an LDAP Search

An LDAP search has the potential to retrieve information about all objects within a specific scope that have certain characteristics - for example, the telephone number of every person in a department.

The following are used to accomplish an LDAP search:

- A *search base* (the distinguished name of the search base object) defines the location in the directory from which the LDAP search begins.
- A *search scope* defines how deep to search within the search base.
  - *Base*, or zero level, indicates a search of the base object only.
  - *One level* indicates a search of objects immediately subordinate to the base object, but does not include the base object itself.
  - *Subtree* indicates a search of the base object and the entire subtree of which the base object distinguished name is the topmost object.
- A *filter* allows certain entries in the subtree and excludes others.
- A *selection* indicates what attributes to return from objects that match the filter criteria.
- Optional controls affect how the search is processed.

Figure 3.1 illustrates the base distinguished name and the search scope of an LDAP search.

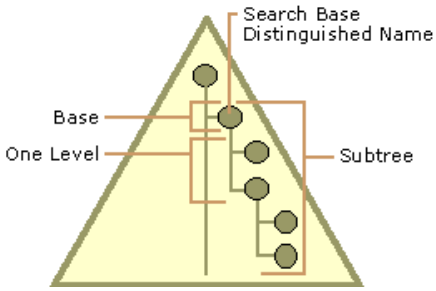
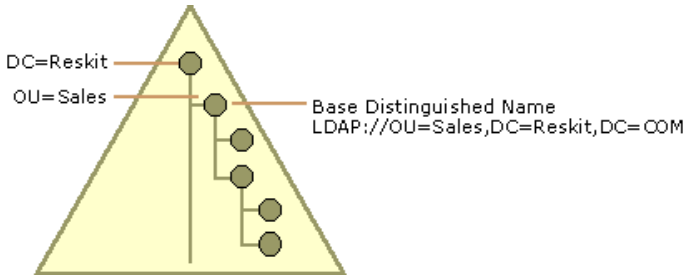


Figure 3.1 LDAP Search Base and Search Scope

Figure 3.2 shows the base distinguished name for a container object.



If your browser does not support inline frames, click here to view on a separate page.

Figure 3.2 Base Distinguished Name for an LDAP Search

Search Filters

By using search filters, you can define search criteria that provide better control to achieve more effective and efficient searches. For example, you might be interested in all the users whose surname is "Smith," or you might want to find out all the team members who report to the manager named "Mary Jones." ADSI supports LDAP search filters as defined in RFC 2254. These search filters are represented by UTF-8 strings. Table 3.1 illustrates some commonly used search filter strings.

Table 3.1 Common LDAP Search Filters

Filter	Description
(objectCategory=*)	All objects.
(&(objectClass=user)(!(cn=susan)))	All user objects except "susan".
(cn=sm*)	All objects with a surname that starts with "sm".
(&(objectClass=contact)(|(sn=Smith)(sn=Johnson)))	All contacts with a surname equal to "Smith" or "Johnson".

The search filters shown in Table 3.1 use one of the following formats:

(<*attribute*><*operator*><*value*>)

- Or -

(<*operator*>(<*filter1*>)(<*filter2*>)...)

Table 3.2 shows some of the most frequently used search filter operators.

Table 3.2 Commonly Used LDAP Search Filter Operators

Operator	Description
=	Equal to
~=	Approximately equal to
<=	Lexicographically less than or equal to
>=	Lexicographically greater than or equal to
&	AND
|	OR
!	NOT

For more information about the LDAP search query syntax and operators, see the Microsoft Platform SDK link and the Request for Comments (RFC) link on the Web Resources page at http://windows.microsoft.com/windows2000/reskit/webresources. On the Request for Comments (RFC) site, follow the links to RFC 2254.

ObjectCategory vs. ObjectClass in a Search Filter

Because of the existence of the class inheritance hierarchy in the schema, every object in Active Directory is in fact a member of many classes - four or five on the average. For this reason, the *objectClass* index is prohibitively large (for example, 4*n*, where *n* is the number of objects in the system). In addition, *objectClass* has poor selectivity for many possible class values. For example, a search filter of (*objectClass=securityPrincipal*) returns every user and group object in the system.

On the other hand, *objectCategory* usually refers to the most specific class in the object's class hierarchy. Although *objectClass* can have multiple values, the attribute *objectCategory* has only one. Every Active Directory object has an *objectCategory* attribute whose value is a *classSchema* object.

Every *classSchema* object has an attribute called *defaultObjectCategory*, which is the object category of an instance of the class if none is specified by the user. For most classes, the *defaultObjectCategory* value is the class itself. In the search filter, you can specify *objectCategory=X*, where *X* is the *ldapDisplayName* of a class, and LDAP automatically expands the filter to *objectCategory=*<*defaultObjectCategory* of class *X*>. The *objectCategory* attribute has a syntax of distinguished name, and LDAP automatically converts the value for *objectCategory* to the distinguished name format. For example, if you use *objectCategory*=contact in the filter, the filter changes to *objectCategory*=cn=person,cn=schema,cn=configuration,dc=<*ForestRootDomain*> ("person" is the *defaultObjectCategory* for the class *contact*).

For more information about class inheritance, see "Active Directory Schema" in this book.

LDAP Referrals

When a requested object exists in the directory but is not present on the contacted domain controller, name resolution depends on that domain controller's knowledge of how the directory is partitioned. In a partitioned directory, by definition, the entire directory is not always available on any one domain controller.

An LDAP referral is a domain controller's way of indicating to a client application that it does not have a copy of a requested object (or, more precisely, that it does not hold the section of the directory tree where that object would be, if in fact it exists) and giving the client a location that is more likely to hold the object, which the client uses as the basis for a DNS search for a domain controller. Ideally, referrals always reference a domain controller that indeed holds the object. However, it is possible for the referred-to domain controller to generate yet another referral, although it usually does not take long to discover that the object does not exist and to inform the client. Active Directory returns referrals in accordance with RFC 2251.

In its Configuration container, every domain controller has information about the other domains in the forest. When an operation in Active Directory requires action on objects that might exist in the forest but are not located in the particular domain that is stored on a domain controller, that domain controller must send the client a message that describes where to go to continue this action - that is, the client is "referred" to a domain controller that is presumed to hold the requested object.

Clients do not need to know the name or location of a child domain in order to contact a domain controller in that domain. They can query the root domain and reach the appropriate domain controller by being referred there. Two situations generate this type of domain controller response:

- The base distinguished name of the operation is not in this directory, but the domain controller has knowledge of another LDAP directory where it might be found (an "external referral").
- The base distinguished name of the operation is in this directory, but the operation requires proceeding into portions of the directory tree that are not stored on this domain controller (a subordinate referral).

Every domain controller contains information (called "knowledge") about how the directory is partitioned, and this information can be used with DNS to find the correct Active Directory domain.

Knowledge References

Active Directory stores information about the existence and location of directory partitions, including the names of the directory partitions, the name of the server that is holding read-only copies (partial directory partitions stored on Global Catalog servers), and the name of the server that is holding writable copies (full directory partitions). Active Directory uses this information (known as "knowledge references") to generate referrals to other domain controllers.

Active Directory uses three kinds of knowledge references to generate referrals to other domain controllers:

- A subordinate reference, which is knowledge of a directory partition (or partitions) directly below a directory partition that is held by the domain controller.
- A cross-reference, which is knowledge of one directory partition and which is stored in a cross-reference object. On a specific domain controller, the combination of all cross-references provides knowledge of all directory partitions in the forest, regardless of their locations in the directory tree.

Note The state of cross-reference knowledge at any specific time is subject to the effects of replication latency.

- A superior reference, which is knowledge of a specifically designated referral location that is used when the domain controller has no knowledge of the search base.

Knowledge references form the glue that holds the pieces of the distributed directory together. Because Active Directory is logically partitioned and directory partitions are the discrete components of the directory that replicate between domain controllers, either all objects in a directory partition are present on a particular domain controller or no objects in the directory partition are present on the

domain controller. For this reason, references have the effect of linking the partitions together, which allows operations such as searches to span multiple partitions.

In Active Directory, referrals are generated when the client requests that the directory locate an object where, based on the position at which the search begins, no copy exists in a local directory partition. When Active Directory can determine definitively that no such object exists in the directory (rather than that it might exist somewhere else even though no copy exists here), instead of sending a referral, the directory returns an error message to the client that no such object exists in the forest.

For more information about replication of directory partitions, see "Active Directory Replication" in this book.

Subordinate References

When a client requests a search, the domain controller searches all objects at or below the search base, within the directory partition that the domain controller holds. If a subtree search has a search base that includes child partitions, the domain controller uses subordinate references to return referrals (called *subordinate referrals*) to these partitions.

Subordinate referrals are returned as part of the data that is returned from the base distinguished name partition. The referral contains the distinguished name of the subordinate directory partition and the access point to which queries can be referred. An access point consists of a DNS name and a port number, which is the information that is required to contact a specific LDAP server. Access points are generated from information contained in the cross-reference object.

Cross-References

Cross-references are stored as directory objects of the class *crossRef* that identify the existence and location of all directory partitions, irrespective of location in the directory tree. Cross-references enable every domain controller to be aware of all directory partitions in the forest, not only the partitions that it holds. Because these objects are stored in the Configuration container, the knowledge that they store is replicated to every domain controller in the forest.

Values for the following attributes are required for each cross-reference:

- *nCName.* The distinguished name of the directory partition that the *crossRef* object references. ("nC" stands for "naming context," which is a synonym for "directory partition.") The combination of all of the *nCName* properties in the forest defines the entire directory tree, including the subordinate and superior relationships between partitions.
- *dNSRoot*. The DNS name of the domain where servers that store the particular directory partition can be reached. This value can also be a DNS host name.

Cross-reference objects are used to generate referrals to other directory partitions in the forest and to external directories.

Cross-reference objects are created in two ways:

- Internally by the system to refer to known locations that are within the forest.
- Externally by administrators to refer to locations that are external to the forest.

Internal Cross-References

An *internal cross-reference* is an object that is created by the system. For every directory partition in a forest, there is an internal cross-reference object in the Partitions container (cn=Partitions,cn=Configuration,dc=*ForestRootDomain*). When you create a new forest, the Active Directory Installation Wizard creates three directory partitions: the first domain directory partition, the configuration directory partition, and the schema directory partition. For each of these partitions, a cross-reference object is created automatically. Thereafter, when a new domain is created in the forest, another directory partition is created and the respective cross-reference object is created. Because these cross-reference objects are located in the Configuration container, they are replicated to every domain controller in the forest, and thus every domain controller has knowledge of the name of every partition in the forest (as well as their superior and subordinate relationships to each other). By virtue of this knowledge, any domain controller can generate referrals to any other domain in the forest, as well as to the schema and configuration directory partitions.

External Cross-References

An *external cross-reference* is a cross-reference object that can be created manually to provide the location of an object that is not stored in the forest. If your LDAP clients submit operations for an external portion of the global LDAP namespace against servers in your forest, and you want your forest's servers to refer the client to the correct location, you can create a cross-reference object for that directory in the Partitions container.

There are two ways that external cross-references are used:

- To reference external directories by their disjoint directory name (a name that is not contiguous with the name of this directory tree). In this case, when you create the cross-reference, you create a reference to a location that is not a child of any object in the directory.
- To reference external directories by a name that is within the Active Directory namespace (a name that is contiguous with the name of this directory tree). In this case, when you create the cross-reference, you create a reference to a location that is a child of a real directory object.

Note An external directory that is stored on a Windows 2000-based domain controller does not require an explicit cross-reference object. Because the domain component (dc=) portions of the distinguished names of all Windows 2000 domains match their DNS addresses and because DNS is the worldwide namespace, all Windows 2000-based domain controllers can generate external referrals to each other automatically.

Creating External Cross-References

The only time you have to create a cross-reference object is when you want to extend a search to a directory outside the forest that is a non-Windows 2000 LDAP directory service. In this case, you can use an LDAP editor, such as ADSI Edit or Ldp, to create objects of the class *crossRef* in the Partitions container that reference external directories.

Note To use ADSI Edit and Ldp, install the Support Tools that are located in the Support\Tools folder on the Windows 2000 Server operating system CD. To install the tools, double-click the **Setup** icon in that folder. For information about installing and using the Windows 2000 Support Tools and Support Tools Help, see the file Sreadme.doc in the Support\Tools folder of the Windows 2000 operating system CD. For more information about using ADSI Edit and Ldp, see *Microsoft Windows 2000 Resource Kit* Tools Help.

When you create a cross-reference object, you must provide the values for three attributes:

cn The name that describes the directory. For example, for the domain noam.reskit.com, your *cn* value might be "noam" or something else that describes that domain, such as "NorthAmerica."

nCName The distinguished name of the domain directory partition to which your cross-reference refers. If the domain name is noam.reskit.com, the value of *nCName* would be dc=noam,dc=reskit,dc=com.

dnsRoot The DNS host name of an LDAP server in the domain that is identified by *nCName* (for example, server1.noam.reskit.com). The value of *dnsRoot* can also be the domain name if you do not want to specify a server.

Note You must be able to resolve ("ping") the name in *dnsRoot*, which does not necessarily name another Windows 2000-based

system; it might be the DNS address of an LDAP server instead of a domain controller. If the directory partition is a Windows 2000 domain from another forest, automatically generated knowledge is usually sufficient and no external cross-reference is required.

You can use either ADSI Edit or Ldp to create cross-reference objects in the Configuration container. However, Ldp requires that you provide the distinguished name of an object and its mandatory and optional attribute names and values when you add the object to Active Directory. For more information about using Ldp, see *Microsoft Windows 2000 Resource Kit* Tools Help.

ADSI Edit provides a convenient graphical user interface for creating cross-reference objects.

To use ADSI Edit to create a cross-reference object

1. In ADSI Edit, expand the Configuration container.
2. Right-click the **CN=Partitions** container, click **New**, and then click **Object**.
3. For **Select a class**, you can create objects of only class *crossRef*, which is already selected. Click **Next**.
4. For the *cn* attribute, in the **Value** box, type a name that describes the location, and then click **Next**.
5. For the *nCName* attribute, in the **Value** box, type the distinguished name for the external domain, and then click **Next**.
6. For the *dnsHostname* attribute, in the **Value** box, type a DNS name for the server that hosts the domain directory partition, or type the domain name.
7. When you are sure that your entries are correct, click **Finish**.

To make use of cross-references, clients must be enabled to follow ("chase") referrals that are returned. Windows Address Book chases referrals by default. In Ldp, you can specify **Chase Referrals** in the search options. When you are using ADSI programmatically (for example, by using Active Data Objects [ADO] to search), you must specify whether to chase referrals. For more information about using ADSI programmatically, see the Microsoft Platform SDK link on the Web Resources page at http://windows.microsoft.com/windows2000/reskit/webresources .

Creating an External Cross-Reference for an External Location

To create a cross-reference to an external directory by referencing an external location, you give the *nCName* attribute a value that is the name of the actual external directory. For example, an external LDAP directory might use X.500 naming (such as o=*Organization Name*,c=*Country/Region*), which would be used for the value of the *nCName* attribute. Queries for this directory must specify the external object by name in the search base distinguished name. A request for a referral to such a location might come in the form of an LDAP Uniform Resource Locator (URL) embedded in an e-mail message or from an application that specifically names the directory distinguished name.

Creating an External Cross-Reference for an Internal Location

If you want a subtree search of a portion of your directory to always include an external LDAP directory that is not a Windows 2000 directory service, you can create a cross-reference to the external directory for an internal location. To create an internal location that references an external directory, give the *nCName* attribute of the cross-reference object a value that is an immediate child object of an existing directory object and that also matches the distinguished name of the external directory. Choose the location according to where you want the external directory to be locatable in Active Directory.

This type of cross-reference is especially useful for smoothly integrating dynamic directories. For example, you might use an instant messaging application such as Microsoft® NetMeeting® conferencing software to publish a list of current or planned conference calls. LDAP is an effective protocol for querying such a published list; however, short-lived, highly volatile data is inappropriate for Active Directory storage. Therefore, you might use an in-memory, nonreplicated LDAP server (one that can store volatile data) at an arbitrary point in the namespace. This "volatile" directory service can then be configured to inhabit a name inside your company's Active Directory namespace and be made available to company users through a cross-reference for an internal location in Active Directory. Users can use this location to find the list of conversations by directory tree navigation.

Suppose that your domain name is reskit.com and you have installed your messaging application on a non-Active Directory-aware LDAP server named vds.it.reskit.com. On that server, you would create a directory for your volatile data, such as cn=conversations,dc=reskit,dc=com. Then, on your Active Directory domain controller, you would create a cross-reference object and use the following attribute values:

- cn=conversation server
- nCName=cn=conversations,dc=reskit,dc=com
- dnsRoot=vds.it.reskit.com

When a user performs a subtree search of dc=reskit,dc=com, the client receives results from the Windows 2000-based domain controllers and also a subordinate referral to the volatile directory service server at vdserver.it.reskit.com, which instructs the client to continue the LDAP search from cn=conversations,dc=reskit,dc=com and below on that server.

Superior References

A superior reference is the distinguished name of a directory partition that is stored in the *superiorDNSRoot* attribute on the *crossRef* object for the forest root domain (the first domain created in the forest). A domain controller uses its superior reference to construct a referral only when a search base does not match any directory partition defined by the cross-reference objects. A superior reference contains no directory tree information; it consists of only an access point to which otherwise unanswerable queries can be referred.

By default, *superiorDNSRoot* does not store a value, but the directory uses the "dc=" components of the search base distinguished name to construct the equivalent of a superior referral. You can use the value in the *superiorDNSRoot* attribute to define a location to send all queries that cannot be resolved.

Ambiguous Name Resolution

Ambiguous name resolution (ANR) is the process of searching for a string value in a set of attributes by using one filter of the form (ANR=*string*).

ANR Attribute Set

By default, the following set of attributes is evaluated when you enter an ANR search string in an LDAP filter:

- *givenName* (first name)
- *sn* (surname, or last name)
- *displayName* (the name given the object when it is created)
- *RDN* (the relative distinguished name of the object)
- *legacyExchangeDN* (for enterprises that have upgraded a Microsoft® Exchange installation to a later version of Exchange that is synchronized with Active Directory, the distinguished name of the old Exchange mailbox that corresponds to the user in Active Directory)

- *physicalDeliveryOfficeName* (for example, Building A, Suite 1234)
- *proxyAddresses* (the collection of e-mail addresses over all e-mail address spaces that the Exchange server knows about)

When the (ANR=*string*) filter is encountered, the filter is expanded to include a search of every attribute in the ANR set.

In Active Directory Users and Computers, you can use the **Filter** or **Find** option (on the shortcut menu or on the toolbar), and select the **Custom** and **Advanced** options to enter an ANR filter. Alternatively, you can use an LDAP editor, such as Ldp. From this LDAP client, you can implement filtered LDAP searches and view the LDAP responses. For more information about using Ldp, see "Active Directory Data Storage" in this book, and see *Microsoft® Windows® 2000 Resource Kit* Tools Help.

ANR Matching of an Embedded Space

For the *givenName* and *sn* attributes, if a space is embedded in the string presented in an ANR filter, the string is split at the first such space and each piece of the string is evaluated separately. This feature enables you to search for a user object by providing the first few characters of the first name (*givenName*) and the first few characters of the last name (*sn*). For example, the filter ANR=dar st finds all objects that have a *givenName* attribute value that begins with "dar" and an *sn* attribute value that begins with "st". In this example, the filter would return a user who has a *givenName* attribute of "Darlene" and an *sn* attribute of "Stuart," as well as a user who has a *givenName* attribute of "Darren" and an *sn* attribute of "Strong."

First/Last and Last/First Evaluation

For the attributes *sn* and *givenName*, when a space is embedded in the string presented in an ANR filter, the filter is expanded to find the values in both respective positions. For example, the filter (ANR=dav st) finds both the user David Strong and the user Steven Davis.

Expanded ANR Filter

When an ANR filter is encountered in an LDAP search, the filter is expanded to construct an OR operation on the string for every attribute in the ANR set. For the *sn* and *givenName* attributes, the first/last and last/first matching are also applied.

The ANR filter of the form (anr= *xxx yyy*) is expanded to the following filter:

```
(| (displayName=xxx yyy*)
(givenName=xxx yyy*)
(physicalDeliveryOfficeName=xxx yyy*)
(proxyAddresses=xxx yyy*)...
(sn=xxx yyy*)
(& (givenName=xxx*)(sn=yyy*))
(& (givenName=yyy*)(sn=xxx*)))
```

The last three lines of the expanded filter are the portion of the filter that evaluates first name and last name.

Adding Attributes to the ANR Set

You can add attributes to the default ANR set by setting a flag on the *attributeSchema* object. By using ADSI Edit, connect to the schema and then open the properties on the attribute that you want to add. Set the *searchFlags* attribute value to a value that represents a bitwise OR operation of 4 and 1 to the existing value. The value **4** adds the attribute to the ANR set; the value **1** indexes the attribute.

For information about setting the *searchFlags* attribute, see the Microsoft Platform SDK link on the Web Resources page at http://windows.microsoft.com/windows2000/reskit/webresources .

Suppressing First/Last and Last/First Functionality

By default, Active Directory is configured to expand the ANR filter to evaluate the positions of two portions of a string that contains an embedded space. In the case of the *sn* and *givenName* attributes, the evaluation also includes checking whether the portion of the string that precedes the embedded space comes before or after the portion of the string that follows the space. If it comes before the portion of the string that follows the space, it is first/last functionality; if it comes after the portion of the string that follows the space, it is last/first functionality.

The *DSHeuristics* attribute on the Directory Service object (cn=Directory Service,cn=Windows NT,cn=Services,cn=Configuration,dc=*ForestRootDomain*) contains a string value that governs the use of first/last and last/first functionality in the first two character positions. The default value of *DSHeuristics* is **00**, which indicates that both functions are enabled. (For all positions, "0" means "perform the default behavior.") The first character in the string governs first/last functionality; the second character governs last/first functionality.

You can modify the first two characters of the string to suppress either one or both functionalities as follows:

10 = Suppress first/last functionality. (This can also be written "1" or "10000" because both mean that only the first character's behavior must be nondefault.)

01 = Suppress last/first functionality.

11 = Suppress last/first and first/last functionality.

Anonymous Queries

If you want users outside the Windows 2000 forest to be able to query Active Directory for white pages-type lookups, you can enable anonymous access to specific directory objects. In Active Directory, anonymous access can be enabled by making objects in a specific container or containers available to the Everyone group.

There are three key requirements for enabling anonymous access to Active Directory:

Server configuration. Read access must be granted to the Everyone group for the containers in Active Directory that are going to allow anonymous access.

Client configuration. An LDAP-compatible client, such as Windows Address Book, must be available to search in Active Directory.

Distinguished name format. Active Directory requires that clients use a search base that contains an LDAP distinguished name (called the "base DN"), which includes both the DNS domain name and the specific container to which anonymous access has been granted.

Note For anonymous access to be available for Internet users, anonymous access must be enabled on the Internet Information Services (IIS) Web server.

Using Access Control to Enable Anonymous Access

You can use access control to allow access to certain objects that you want to make available to anonymous users. You can do so by granting read access to the Everyone group for a container object that stores the public objects.

Members of the Everyone Group

In Active Directory, users who log on with authentication are automatically included in the Authenticated Users group. Users who log on without authentication are represented as Anonymous Logon. Users who log on as Guest are included in the Domain Guests group. In all cases, the users are members of the Everyone group. Therefore, providing access to the Everyone group covers all potential anonymous users in addition to all authenticated users.

Note On all computers that run Windows 2000 or Windows NT, there is a built-in Guest account, which does not require a password (the password can be blank) and is meant to be shared by users who do not have personal accounts. These users are, in a sense, anonymous users. The Guest account is disabled by default.

By default, the Everyone group has read access to the domain object and its properties, but the access is not inherited by child containers. This access is enabled so that a user can log on anonymously in the event the user's password expires. When a password expires, the password must be changed before the user can log on. To change the password, the user must connect to the domain and change the password by providing the old password and the new password. Because the password has expired, this operation can proceed only if the user is able to connect anonymously.

Caution Avoid granting anonymous access to the domain-level container at any level other than "this object only," which is the default setting. By using this default setting, you enable all users to read the properties of the domain object itself but not see any of the objects below it in the hierarchy.

Assigning Read Access for Everyone

By granting the Everyone group read access to a specific container of objects, you can enable anonymous access to only that portion of Active Directory. You can use Active Directory Users and Computers to assign access control to a container in which you have placed the user objects that you want to make available for public access. To see the security options, enable **Advanced Features**.

To enable anonymous access to an Active Directory container

1. In the Active Directory Users and Computers console, if **Advanced Features** is not enabled, on the **View** menu, click **Advanced Features**.
2. Right-click the container to which you want to provide anonymous access.
3. Click **Properties**, click the **Security** tab, and then click **Advanced**.

   In the **Permission Entries** box, if the Everyone group is not listed, click **Add**. In the **Name** column, click **Everyone**, and then click **OK**.
4. In the **Permission Entry for *ContainerName*** dialog box, click the **Properties** tab.
5. In the **Apply onto** list, click **User objects**.
6. In the **Permissions** list, in the **Allow** column, click the permission or permissions that you want to allow (for example, **Read General Information**). Then click **OK**.
7. On every security warning message that appears, if any, click **Yes**.
8. In the **Access Control Settings** dialog box, click **OK**.

Caution Enabling anonymous queries weakens the inherent security in an Active Directory environment. Special care should be taken when you are deciding what containers and attributes are to be exposed to anonymous users.

Granting Read All Properties for Anonymous Queries

The built-in group Pre-Windows 2000 Compatible Access has Read All Properties access on user and group objects. By default, the Everyone group is not a member of this group. If you want to grant this level of access to anonymous users, you can add Everyone to this group. By doing so, you allow anonymous read access to all properties of all user and group objects. For more information about access control, see "Access Control" in this book.

Security Precautions for Anonymous Access

Any time that anonymous access is enabled where Internet access is available, it is critical to domain security that firewalls be configured to protect the ports that are used to gain entry to Active Directory.

A firewall is a combination of hardware and software that provides a security system, usually to prevent unauthorized access from the Internet to an internal network. A firewall prevents direct communication between network and external computers by routing communication through a proxy server outside the network. The proxy server determines whether it is safe to let a file pass through to the network.

Firewalls should be configured to protect the following ports:

- Port 389 for LDAP
- Port 636 for LDAP over Secure Sockets Layer (SSL)
- Port 3268 for the Global Catalog
- Port 3269 for the Global Catalog over SSL

For more information about configuring firewalls, see "Internet Protocol Security" in the *TCP/IP Core Networking Guide*.

Global Catalog and LDAP Searches

The Global Catalog enables searching for Active Directory objects in any domain in the forest without the need for subordinate referrals, and users can find objects of interest quickly without having to know what domain holds the object.

Global Catalog Servers

A Global Catalog server is a domain controller that stores extra information; its database stores rows for every object in the forest instead of rows for only the objects in one domain. The rows that store objects that occur in domain directory partitions other than the local domain partition hold only a subset of attributes for each object. In this way, the Global Catalog enables forest-wide searches without requiring replication of the entire contents of Active Directory to every domain controller. The Knowledge Consistency Checker (KCC) process creates a replication topology that ensures delivery of the contents of every directory partition to every Global Catalog server in the forest.

Note A Global Catalog server stores full (writable) copies of the schema and configuration directory partitions - the same as any domain controller.

By default, the server on which you install Active Directory to create the first domain in a new forest is a Global Catalog server. Thereafter, you must designate additional Global Catalog servers, if they are needed.

Searching the Global Catalog vs. Searching the Domain

The decision whether to search the Global Catalog or the domain is based on the scope of the search:

- When the scope of a search is the domain or an organizational unit, the query can be resolved within the domain partition by using

an LDAP search.

- When the scope of a search is the forest, the query can be resolved within any partition by using a Global Catalog search.

Searches That Use the Global Catalog by Default

Any time that you specify port 3268, you are searching in the Global Catalog. In addition, the Global Catalog is searched by default under the following conditions:

- During the logon process when a user principal name is presented. The Global Catalog is searched to find the domain and account name on the basis of the user principal name.
- During the logon process to expand universal groups. Universal group membership can span domains. It is possible, therefore, that a user has a membership in a universal group that is not in the logon domain. For this reason, the Global Catalog is contacted to search the membership of universal groups. If a membership is found, the group is attached to the user's logon credentials.
- When you choose **Entire Directory** in a search-scope list.
- When you write the value for a distinguished name-valued property, where the distinguished name represents a nonlocal object. For example, if the member that you are adding is from a different domain, the Global Catalog is used to verify that the user object represented by the distinguished name actually exists.

Global Catalog Search Base

For an LDAP search, you must supply a valid base distinguished name. For a Global Catalog search, the base distinguished name can be any value, including the value "NULL" (" "). A base distinguished name of NULL effectively scopes the search on the search computer to the Global Catalog. If you use a NULL base distinguished name with a scope of one level or subtree and specify port 389 (the default LDAP port), the search fails. Therefore, if you submit a NULL search to the Global Catalog port and then change the port to the LDAP port, you must change the base distinguished name for the search to succeed.

Note

Windows Address Book is configured automatically with the value "NULL" for server name, account name, and base distinguished name. The default port is port 3268; so to submit the search to port 389, you must provide a valid base distinguished name as defined in RFC 2247. A blank base distinguished name fails on either port.

Characteristics of a Global Catalog Search

The following characteristics differentiate a Global Catalog search from a standard LDAP search:

- Global Catalog queries are directed to port 3268, which explicitly indicates that Global Catalog semantics are required. By default, ordinary LDAP searches are received through port 389. If you bind to port 389, even if you bind to a Global Catalog server, your search includes a single domain directory partition. If you bind to port 3268, your search includes all directory partitions in the forest. If the server you attempt to bind to over port 3268 is not a Global Catalog server, the server refuses the bind.
- Global Catalog searches can specify a non-instantiated search base, indicated as "com" or " " (blank search base).
- Global Catalog searches cross directory partition boundaries. The extent of the LDAP search is the directory partition.
- Global Catalog searches do not return subordinate referrals. If you use port 3268 to request an attribute that is not in the Global Catalog, you do not receive a referral to it. Subordinate referrals are an LDAP response; when you query over port 3268, you receive Global Catalog responses, which are based solely on the contents of the Global Catalog. If you query the same server by using port 389, you receive referrals for objects that are in the forest but whose attributes are not referenced in the Global Catalog.

Note An external referral can be returned by the Global Catalog if a base-level search for an external directory is submitted and if the distinguished name of the external directory uses the domain component (dc=) naming attribute. This referral is returned according to the ability of Active Directory to construct a DNS name from the domain components of the distinguished name and not based on the presence of any cross-reference object. The same referral is returned by using the LDAP port; it is not specific to the Global Catalog. (For more information about constructing a DNS name from the domain components, see "Superior References" earlier in this chapter.)

Effect of Global Catalog When Searching Back Links and Forward Links

Some Active Directory attributes cannot be located specifically by finding a row in the directory database. A back link is an attribute that can be computed only by referencing another attribute, called a forward link. An example of a back-link attribute is the *memberOf* attribute on a user object, which relies on the group attribute *members* to derive its values. For example, if you request the groups of which a specific user is a member, the forward link *members*, an attribute of the group object, is searched to find values that match the user name that you specified.

Because of the way that groups are enumerated by the Global Catalog, the results of a back-link search can vary, depending on whether you search the Global Catalog (port 3268) or the domain (port 389), the kind of groups the user belongs to (global groups vs. domain local groups), and whether the user belongs to groups outside the local domain. Connecting to the local domain does not locate the user's group membership in groups outside the domain. Connecting to the Global Catalog locates the user's membership in global groups but not in domain local groups because local groups are not replicated to the Global Catalog. For more information about searching on back-link attributes, see "Active Directory Data Storage" in this book.

Searching for Deleted Objects

When an Active Directory object is deleted, it is stored in the Deleted Objects container for a configurable period of time to allow replication of the deletion to occur. By using the **Show Deleted Object** control (controlType = 1.2.840.113556.1.4.417), in conjunction with search commands, you can view Active Directory objects that have been deleted but not yet garbage collected. These objects are called *tombstones*. After they are deleted by garbage collection, they no longer exist in the directory database.

To retrieve tombstone objects, list the contents of the Deleted Objects container. You can use Ldp to find these objects by using an LDAP control.

To use Ldp to search the domain for deleted objects (tombstones)

1. On the **Start** menu, click **Run**, and then type **ldp**.
2. Connect and bind to a domain controller in the domain whose tombstones you want to retrieve.
   - To connect, on the **Connection** menu, click **Connect**, and then type a server name and a port number.
   - To bind, on the **Connection** menu, click **Bind**, and then type an account name, password, and domain if you want to connect to a domain other than the domain to which you are currently logged on.
3. On the **Browse** menu, click **Search**.
4. In the **Search** dialog box, for **Base DN**, type the distinguished name of the domain whose tombstones you want to retrieve.
5. In the **Filter** box, use the filter **(isDeleted=*)**.
6. Under **Scope**, click **Subtree**.
7. Click **Options**.

8. In the **Search Options** dialog box, under **Search Call Type**, click **Extended**.
9. Click **Controls**. Then in the **Object Identifier** box, type the following:
10. Under **Control Type**, click **Server**.
11. To add the control to the **Active Controls** list, click **Check in**. Then click **OK**.
12. In the **Search Options** dialog box, click **OK**.
13. In the **Search** dialog box, click **Run**.

For more information about how to use Ldp, see *Microsoft Windows 2000 Resource Kit* Tools Help. For more information about using Ldp for directory management and troubleshooting tasks, see "Active Directory Diagnostics, Troubleshooting, and Recovery" in this book.

LDAP Search Clients

Several clients that are available with Windows 2000 Server provide varying degrees of sophistication for searching Active Directory.

Administrative Clients

Administrative clients such as the Active Directory Users and Computers MMC snap-in provide search and filter options when certain objects are selected. In addition, when you open Network Places, Entire Network, or Directory, the **Find** option that is available provides the same search capabilities as the **Find** option in Active Directory Users and Computers.

Using the Filter Options Command in Active Directory Users and Computers

In Active Directory Users and Computers, you can use filter options to define the information that you want to view. When you apply a filter, only the objects you specify in the filter are displayed in the filtered container. The default filtering option displays all types of objects (that is, no filter is applied). However, it is possible to select only certain types of objects to be displayed, such as users, groups, contacts, and so on. Also, you can customize the kind of information that is displayed within each object type by selecting fields and specifying a condition and value, or by entering an LDAP query. The filter remains in effect until you remove it. It is not displaced or overridden by any other filter.

Active Directory Users and Computers provides options for filtering that do not require you to create an LDAP query. (For more information about using standard filter options, see Windows 2000 Server Help.)

However, if the options in the standard filter user interface do not meet your needs, you can use advanced (customized) filter options to write an LDAP query that does. For example, the standard filter user interface allows you to select "users" as the objects to filter, but it does not provide the ability to specify all possible attributes for a user. For example, if you want to display only the user accounts that were created after a specific date, you can use an LDAP filter to retrieve only these users by using the *whenCreated* attribute value in an LDAP filter.

To use Filter Options to apply a filter to a container by using an LDAP query

1. In Active Directory Users and Computers, in the console tree pane, click the container for which you want to filter objects.
2. On the **View** menu, click **Filter Options**.
3. In the **Filter Options** dialog box, click **Create custom filter**, and then click **Customize**.
4. Click the **Advanced** tab.
5. In the **Enter LDAP query** box, type an LDAP query string, for example:

   (&(objectCategory=user)(whenCreated=991122000000Z))

   Note The time format YYMMDDHHMMSSZ must be used to represent the two-digit year (YY), month (MM), day (DD), hour (HH), minutes (MM), and seconds (SS) and must end with an uppercase "Z". You can use zeros to fill in the time elements if you are not interested in the time of creation.
6. Click **OK** twice. Double-click the container to view the filtered objects.

A filter remains in place until you remove it. You can remove a filter in Active Directory Users and Computers by clicking the **Filter** icon on the toolbar and then, in the **Filter Options** dialog box, click **Show all types of objects**.

Using the Find Command in Active Directory Users and Computers

You also can use an LDAP query to search a container without applying a filter to the container. To create an LDAP query to display only specific objects in a container, use the **Find** option on the container shortcut menu, as described in the following procedure.

To use Find to search a container by using an LDAP query

1. In Active Directory Users and Computers, right-click the container you want to search, and then click **Find**.
2. In the **Find** box, click **Custom search**, and then click the **Advanced** tab.
3. In the **Enter LDAP query** box, type an LDAP query string, for example:
4. (&(objectCategory=user)(whenCreated=991122000000Z))
5. Click **Find Now** to display the search results.

Note In advanced filters, you can use matching rules to implement search flags if you know the correct LDAP control object identifier (also known as an "OID") value to use and how to compute the value. For example, you can search on the *userAccountControl* attribute to specify users who have disabled accounts, or you can search on the *groupType* attribute to find all the Global groups in a search base. For more information about using search flags and matching rules, see the Microsoft Platform SDK link on the Web Resources page at http://windows.microsoft.com/windows2000/reskit/webresources .

Windows Address Book

Windows Address Book is a generic LDAP search client that is designed to work with any LDAP server. Address Book is integrated into the Windows 2000, Microsoft® Internet Explorer version 4.0 and later, and Windows 95 and Windows 98 shells to provide the capability to search for people in one or more directory services, including Active Directory. Address Book version 5.0 is included with Windows 2000 and Microsoft® Internet Explorer 5, and provides a set of accounts that are preconfigured to enable easy access to information in several Internet "white pages" directories, such as InfoSpace and VeriSign.

Address Book Access to Active Directory

In Windows 2000, Address Book provides access to Active Directory as follows:

- It is automatically configured to search the Global Catalog of the forest to which the user is bound when the user selects **Search** and then **Find People** on the **Start** menu.
- It supports UTF-8 to expose Unicode characters, which enables customers to search for users and resources whose names contain non-ASCII characters. This feature is important in European and Asian countries/regions.

- By using the property access control lists (ACLs) on an object, it can display an edit box if the user has permissions to modify the property.
- It provides flexible matching by using support for ambiguous name resolution.
- It can gain access to properties of objects that are created from extended object classes.
- It exposes an API for processing LDAP URLs. Address Book is included with Internet Explorer and registers itself as an LDAP URL (ldap://) handler.
- It supports chasing LDAP referrals (RFC 2251) when the Address Book searches in Active Directory over port 389.

By default, a server name and account name of NULL are configured for Address Book. Active Directory dynamically provides the server name that is cached during domain controller location for the server name, and it uses the logon name of the authenticated user as the account name. The Active Directory properties in Address Book show the default settings.

To view Active Directory properties in Address Book

1. On the **Start** menu, point to **Programs**, point to **Accessories**, and then click **Address Book**.
2. On the **Tools** menu, click **Accounts**.
3. Click **Active Directory**, and then click **Properties**. The property sheet displays the directory service account settings, as shown in Figure 3.3.

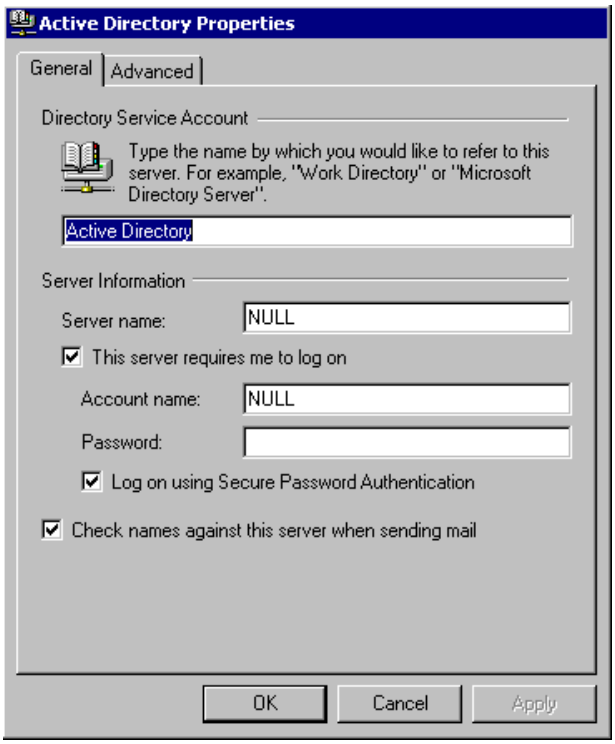


Figure 3.3 Server Settings in the Active Directory Properties Dialog Box in Address Book

Search Base in Address Book Searches

As an LDAP directory, Active Directory requires an RFC 2247-compliant distinguished name, or search base, to perform an LDAP search. By default, a search base distinguished name of NULL is configured in Address Book. During domain controller location, the Locator caches the DNS name of the found domain controller. When requesting a search of Active Directory, Address Book uses as the search base the cached distinguished name of the domain in which the logon account was authenticated.

The **Advanced** tab in the **Active Directory Properties** dialog box displays the settings that determine how Address Book searches are performed, as shown in Figure 3.4.

Note By default, the port setting identifies the Global Catalog port 3268. Clicking **Use Default** changes the port to the default LDAP port 389.

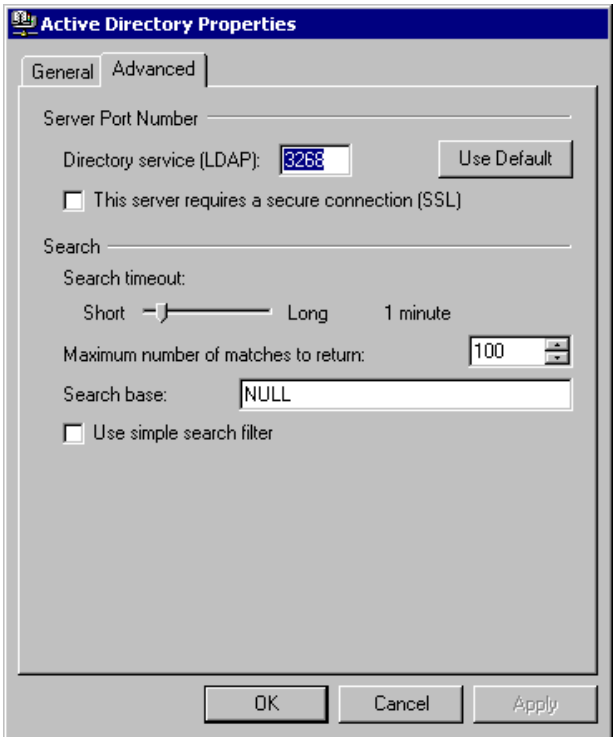


Figure 3.4 Advanced Search Settings in the Active Directory Properties Dialog Box in Address Book

Active Directory Availability on Windows 98 and Windows NT 4.0 Clients

Computers that are running Windows 98 or Windows NT 4.0 that have Internet Explorer 5 installed are not able to gain access to Active Directory unless the clients are configured with a server name and search base and, if the server requires an authenticated logon, an account name (for example, *domainName\userName*). These values must be entered in the **General** and **Advanced** tabs to define Active Directory as an LDAP server for Address Book.

For instructions about how to change the Address Book settings, see Address Book Help. For more information about domain controller location, see "Locating Active Directory Servers" earlier in this chapter.

Ldp

Ldp is a tool that you can use to search in Active Directory by using LDAP filters. You also can use Ldp to add, delete, and modify objects in Active Directory and to perform extended LDAP operations by using LDAP controls. To use Ldp, install the Support Tools that are located in the Support\Tools folder on the Windows 2000 Server operating system CD. To install the tools, double-click the **Setup** icon in that folder. For information about installing and using the Windows 2000 Support Tools and Support Tools Help, see the file Sreadme.doc in the Support\Tools folder of the Windows 2000 operating system CD. You can run the Ldp tool from the **Start / Run** menu, or from the command line by typing **ldp**.

For more information about how to use Ldp, see Ldp Help in *Microsoft Windows 2000 Resource Kit* Tools Help. For more information about using Ldp for directory management and troubleshooting tasks, see "Active Directory Diagnostics, Troubleshooting, and Recovery" in this book.
