

*Mastering the Domain Name System*



# DNS

*on Windows Server 2003*

**O'REILLY®**

*Cricket Liu, Matt Larson & Robbie Allen*

## CHAPTER 8

---

# Integrating with Active Directory

*“The face is what one goes by, generally,” Alice remarked in a thoughtful tone.*

With the release of Windows 2000, Microsoft replaced the Windows NT Security Account Manager (SAM) with Active Directory (AD), which serves as the repository for information about users, groups, computers, and other network resources. In contrast to the SAM, Active Directory is built on several well-known standards including the Lightweight Directory Access Protocol (LDAP) for accessing and manipulating data, Kerberos for authentication, and—you guessed it—DNS for name resolution.

In fact, using DNS for name resolution is one of the major improvements of Active Directory over Windows NT, which relied on the Windows Internet Naming Service (WINS). Microsoft made the decision to develop WINS in the early days of Windows NT because, at the time, DNS did not support dynamic update capability, which Microsoft needed for its clients. As a result, many companies had to implement both services: DNS for standard Internet-based name resolution and WINS for the Windows NT environment. This often pitted the NT administrators against the DNS administrators because of the need to maintain two separate namespaces. Over time, dynamic update support was added to DNS, and WINS failed to garner industry support—in no small part because it was a proprietary Microsoft offering.

Even with the opportunity to get rid of WINS, migrating to Active Directory hasn't always resulted in a harmonious union between AD and DNS administrators. While Windows NT had virtually no DNS requirement, Active Directory is at the opposite extreme. It is completely dependent on DNS. If DNS becomes unavailable, clients may fail to authenticate or log in to Active Directory, and domain controllers will not be able to replicate changes throughout the forest. This highly visible dependency on DNS requires that the AD and DNS administrators work closely together (assuming they are in separate groups) and agree on implementation details, which can sometimes be a challenge. It is not uncommon for DNS administrators to be reluctant to delegate part of the namespace for Active Directory, and AD administrators are often

hesitant to entrust a critical component to another group and forgo the advantages of AD-integrated DNS.

This chapter explores many of the key DNS-related issues you need to be aware of when implementing and supporting Active Directory. We cover how Active Directory uses DNS for service advertisement and domain controller location; and, conversely, how Active Directory can be used to enhance DNS by providing robust replication and security for zone data. We do not—in fact, *cannot* in a single chapter—cover the numerous other Active Directory components. For more information on designing, implementing, and automating Active Directory, see *Active Directory*, Second Edition (O’Reilly) by our own Robbie Allen. For examples on how to perform common Active Directory administrative tasks, see *Active Directory Cookbook* (O’Reilly), also by Robbie.

## Active Directory Domains

One of the first issues you have to consider when implementing an Active Directory infrastructure is how many domains you need and what to name them. Active Directory domain names are DNS domain names, but—and this is important—not every DNS domain name is an Active Directory domain name.\* So while an organization’s Active Directory namespace resembles its DNS namespace, the two don’t have to be identical.

The number of domains you create in your forest is largely dependent on your administrative and replication requirements. A domain is mastered by one or more domain controllers, which are servers that have writeable copies of the data (about users, groups, computers, etc.) contained in the domain. Unfortunately, Active Directory is not like DNS, where a single name server can be authoritative for multiple zones. A domain controller can be authoritative only for a single Active Directory domain. To create a new Active Directory domain, you have to install a new domain controller—your existing domain controllers cannot be used. However, Active Directory uses a multimaster replication system, unlike DNS, and consequently any domain controller can process updates and replicate the changes to the other domain controllers in the domain.

## Domains, Domain Trees, and Forests

Domain trees and forests are two important Active Directory concepts. A *domain tree* is simply a collection of one or more domains that share a common namespace. The *fx.movie.edu* and *movie.edu* domains would be considered part of the *movie.edu*

\* And every square is a rectangle, but not all rectangles are squares. All registered mail is certified, but not all certified mail is registered. You get the idea.

domain tree; however, the *example.com* domain, if created after *movie.edu*, would be in a separate domain tree called *example.com*. If the domain you create does not contain the full name of the parent domain or forest root domain, it is considered part of a separate domain tree.

A *forest* is a collection of one or more domain trees. The domains in the *movie.edu* domain tree and the *example.com* domain tree could be part of the same forest. A domain tree is based on a common namespace, but a forest is not.

A forest is named after the first domain created in the forest. If *movie.edu* was the first domain we created, the forest is automatically named *movie.edu*. We can then create additional domains for *fx.movie.edu* and *example.com* all belonging to the *movie.edu* forest. Another option is to create the *example.com* domain in its own forest, which has certain implications for user access.

All domains within a forest, regardless of which domain tree they are part of, are trusted by each other from an authentication and authorization perspective. For this reason, the forest is considered the primary *security boundary* in Active Directory. By making the *example.com* domain part of the *movie.edu* forest, users in *example.com*, by default, have read access to much of the information in the *movie.edu* domains, and vice versa. Also, users in *example.com* can have control delegated to them over objects in the *movie.edu* domains and vice versa. If *example.com* is created in a separate forest, the users in that domain do not have access to the *movie.edu* forest, and they cannot have access delegated to it (unless a special cross-forest trust is created between the two forests).\*

## Domain Models

Two Active Directory domain models are commonly used. Your choice of domain model is important because it dictates DNS requirements, as we discuss later. One model consists of an empty root domain with geographic or organizational subdomains. We recommend that you do not create organizational-based subdomains if you can help it since they are very likely to change at some point.†

Some of the benefits of the multidomain model include the ability to delegate administration for management of portions of a forest, greater control over how data is replicated within the forest, and decreased exposure to problems that can impact a single domain. For example, if an application went out of control and mistakenly started creating tons of objects in a domain, at some point the hard disk on the

\* Windows Server 2003 supports a new trust type called forest trust, which allows you to have a single transitive trust that spans two forests. If you want to understand more about how trusts work, get *Active Directory* (O'Reilly).

† Fortunately, a new feature of Windows Server 2003 allows you to rename domains, but the process is arduous. For more information on renaming domains, see <http://www.microsoft.com/windowsserver2003/downloads/domainrename.mspx>.

domain controllers in that domain would fill up. The domain would then enter an unstable and possibly unusable state. However, the domain controllers in other domains would continue to function without much impact. This example is admittedly contrived, but it shows you the benefits of autonomy in the multidomain model.

The downsides of this model include the increased support costs of setting up additional domain controllers—remember that each domain must have its own set of domain controllers—and additional domain configuration (configuring security, group policy objects, organizational units, etc.). Also, finding data in a multidomain forest is not always easy. Since data can be spread across several domains, you have to query the global catalog to perform forestwide searches. The global catalog contains a read-only copy of the objects in all domains in the forest. The drawback to the global catalog is that it contains only a subset of the attributes of objects. If the global catalog doesn't contain the attribute you want, you have to perform an additional query against the domain the object resides in after you've queried the global catalog.

The other common domain model is simply a single domain. Over time, many organizations have come to recognize the increased support costs associated with supporting additional domains and have found that their initial reasons for needing multiple domains may no longer be valid. With a single domain, you typically need fewer domain controllers, which results in decreased costs. Also, the global catalog does not play as big a role because you do not need to search across multiple domains.

The downside to the single domain model is decreased flexibility in replicating data. Since all of your objects are in a single domain, wherever you need to deploy a domain controller, you have to replicate all of the objects in the domain to it. This type of model does not lend itself well to branch office deployments where you have a lot of domain controllers, many of which may have slow WAN connections.

There are variations to these models, but this should give you the general idea. A good rule of thumb is the fewer domains the better. And if you can get by with a single domain, that's great.

## Three Options for the Root Domain Name

Once you've decided on a domain model, you need to choose a name for your root domain, also known as the forest root domain. The root domain name is very important because it determines which name servers can be authoritative for the corresponding DNS namespace. You have three basic options for naming your root domain: use the same name as an existing DNS domain, create a new subdomain from an existing domain, or use a name that doesn't correspond to any of your DNS domains (i.e., a disjoint or private namespace). Each option has minor advantages

and disadvantages based on your environment, but, as with most naming convention discussions, the decision is largely subjective.

### Same name as an existing DNS domain

Consider Movie University. The apex (or top) of Movie U.'s DNS namespace is *movie.edu*, with subdomains named *fx.movie.edu*, *classics.movie.edu*, and *comedies.movie.edu*. This namespace is represented in Figure 8-1.

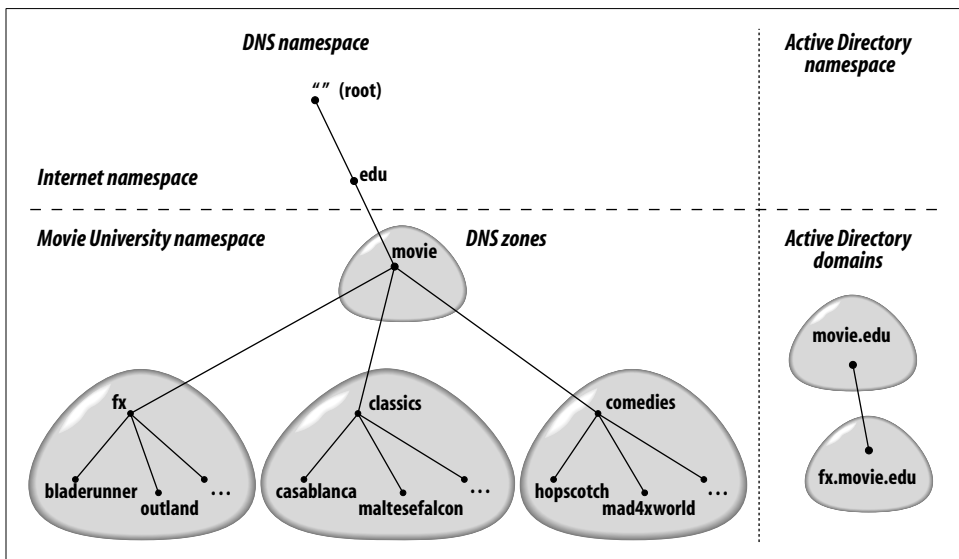


Figure 8-1. Movie University's namespace

We could root Active Directory's namespace at the root of Movie U's DNS namespace. That would result in *movie.edu* being the forest root domain. If we decide that the special effects lab needs a dedicated Active Directory domain, we can create the *fx.movie.edu* domain as a child domain in the *movie.edu* forest. Everyone else at Movie U. can be part of the *movie.edu* Active Directory domain, even though individual hosts may fall into different DNS domains. If we did nothing more, the resource records needed by Active Directory would be added to the *movie.edu* and *fx.movie.edu* zones.

If your authoritative name servers are not domain controllers and you want to use AD-integrated zones (more on this later), or you want some other name servers to be authoritative for the zone that contains the Active Directory resource records, you have to create delegations. All of the required Active Directory-specific resource records are stored in subdomains of the DNS domain whose name corresponds to the AD domain. These subdomains are named *\_msdcs* (e.g., *\_msdcs.movie.edu*), *\_tcp*, *\_udp*, *\_sites*, and *DomainDNSZones*. The forest root domain will also have a

*ForestDNSZones* subdomain.\* To delegate the Active Directory DNS responsibilities when the Active Directory domains are named after an existing DNS domain, you need to delegate those subdomains. In turn, those subdomains become zones on the delegated servers.

### Subdomain of an existing DNS domain

Now let's say that Movie U. wanted to completely separate its AD namespace from its DNS namespace. Another common approach is to create a subdomain of an existing domain, such as the top-level domain for the organization. For example, we could name the forest root domain *ad.movie.edu* or *corp.movie.edu*. If we still needed to create a subdomain for the special effects lab, it could be named *fx.ad.movie.edu*.

This model is the easiest to implement if you want to delegate responsibility for the Active Directory–critical zones to name servers other than the main set of authoritative name servers for your organization. Instead of delegating subdomains as in the previous model, you need to delegate only the zone with the same name as the forest root domain (e.g., *ad.movie.edu*) from your main forward-mapping zone (e.g., *movie.edu*).



Microsoft recommends this option for most deployments and uses it for its own corporate Active Directory deployment.

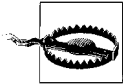
### Disjoint or private name

The last major option for naming your forest root domain is not to base it on any of your top-level domain names and use a disjoint or private name. The most common suffix that is used in this scenario is *.local* (e.g., *movie.local*). This model is typically chosen by organizations that do not want their Active Directory DNS namespace to be public.

In nonproduction forests, we recommend following the guidelines in RFC 2606 and using *.test* or *.example* instead of *.local*, which are suffixes reserved for testing purposes. If you need to create a lab or test environment that your general community does not need to access, this is a good model to use. In fact, you can create a forest without requiring any delegations at all from your main forward-mapping zone, assuming you have control over the resolver configuration for the clients in the lab. To do this, you would need to create the *movie.test* forest, enable the DNS server on one or more domain controllers in the domain, and make the *movie.test* zone AD-integrated.

\* The *DomainDNSZones* and *ForestDNSZones* domains are new in Windows Server 2003. They are used to support application partitions, described later in this chapter.

In order for clients to resolve DNS names outside of the forest, you would need to point the resolvers on the domain controllers to each other, and configure forwarders to forward unresolved requests to your organization's main authoritative name servers. At that point, all you need to do is point the resolvers of the clients that are going to use the *movie.test* forest to one of the domain controllers that are running the Microsoft DNS Server in *movie.test*. Now you have a fully functional Active Directory forest that didn't require any delegations from your main forward-mapping zone.



Microsoft strongly discourages using a single-label domain name, such as *movie*, for your forest root domain name. If you do, it requires additional configuration on your servers. Microsoft's reasoning behind this is to limit the number of misconfigurations that result in hammering the top-level root name servers with unnecessary DNS requests.

## Storing Zones in Active Directory

One of Microsoft's innovative uses of Active Directory is for storing (and replicating) DNS zone data. A traditional name server stores copies of the zones it supports in files on a local disk. In this model, you have a primary master name server that replicates the zone data to secondary name servers. A secondary can process updates to a zone from its master name server in two different ways. The original method supported by DNS is zone transfer, which allows secondary name servers to request a full copy of a zone. A newer method, which is an improvement on the zone transfer process, is incremental zone transfer. With incremental zone transfer, a secondary name server can request just the updates to the zone that occurred since its last transfer.

Active Directory provides another method for replicating zone content, albeit only for name servers running on domain controllers. You can make a zone *AD-integrated*, which means that instead of storing zone content in text files, it is stored in the Active Directory database. This makes a lot of sense because you take advantage of Active Directory's multimaster replication scheme, which means that any domain controller that is also a primary name server for the AD-integrated zone can update it directly, like a primary name server. With AD-integrated zones, replication is handled automatically, so you don't need to develop your own zone transfer replication topology.

One other note about Active Directory-integrated zones: strictly speaking, you don't have to make every domain controller into a name server for a zone that's AD-integrated within that domain. Since all authoritative servers can be configured to allow zone transfers, a server that loads a zone from Active Directory happily responds to allowed zone transfer requests. So you could conceivably make only a zone's primary master name server AD-integrated and have the secondaries continue

to load from the primary, provided those secondaries are not domain controllers of the domain. However, this defeats one of the purposes of Active Directory integration—letting Active Directory handle zone replication. The other huge advantage of Active Directory integration is secure dynamic updates, which we'll cover in detail shortly.

## The Impact on Replication

While AD-integrated zones have many advantages, the one potential drawback is how zone data gets replicated in Active Directory. Under Windows 2000, AD-integrated zones are stored in the System container in a domain. That means that every domain controller in that domain replicates that zone data regardless of whether the domain controller is a name server or not. For domain controllers that are not name servers, there is no benefit to replicating the data, which results in needless replication traffic. This can be a serious issue for those that have large zones with many domain controllers that are not name servers. Fortunately, a new feature in Windows Server 2003 called application partitions provides a better alternative.

Another issue with Windows 2000 AD-integrated zones is that domain controllers in subdomains could not replicate the forest root zone resource records through AD replication. You would have to configure the subdomain domain controller as a secondary and enable zone transfer. The requirement to replicate forest root records is common in branch office deployments where a certain amount of autonomy is needed due to potential network outages. Application partitions also help with this requirement.

## Using Application Partitions

Active Directory segregates data into one of three partitions: schema, configuration, or domain. Partitions are used to organize and replicate data with a similar scope (i.e., forestwide or domainwide) among domain controllers. Conceptually, partitions are similar to zones. Zones are also used to segregate and replicate data (using zone transfer) among name servers.

In Windows Server 2003, Microsoft added a new type of partition called an *application partition*. Whereas the default partitions have a predefined replication scope, application partitions can be configured to replicate with any domain controller in a forest. Domain controllers that are configured to contain replicas of an application partition become the only servers that replicate the data contained within the partition. Application partitions are not limited by domain boundaries. You can configure domain controllers in completely different domains to replicate an application partition. For these reasons, application partitions make a lot of sense for storing AD-integrated zones. You no longer have to store zone data within the domain partition and replicate it to every domain controller in the domain, even if only a few are

name servers. With application partitions, you can configure Active Directory to replicate DNS data only between the domain controllers running the DNS server in any domain of the forest. To help facilitate the transition from domain-based storage of zones to application partitions, Microsoft provides a couple of default DNS application partitions: one default DNS application partition for each domain in a forest and one for the forest itself. During the installation of a new Windows Server 2003 Active Directory forest, these partitions are created automatically and AD-integrated zones are stored there. If you upgrade from Windows 2000, the default DNS application partitions are still created automatically, but existing AD-integrated zones are not moved into them. You have to do that manually.

You are not required to use the default DNS application partitions. You can create your own or continue to use the System container in the domain partition, which is the only option under Windows 2000. The storage options are summarized in Table 8-1.

Table 8-1. Storage options for AD-integrated zones

| Location  | Replication scope   |
|---|---|
| <i>cn=System,DomainDN</i><br>Example:<br><i>cn=System,dc=movie,dc=edu</i>                       | All domain controllers in the domain, regardless of whether they are a name server. This is the only storage method available under Windows 2000. |
| <i>dc=DomainDnsZones,DomainDN</i><br>Example:<br><i>dc=DomainDnsZones,dc=fx,dc=movie,dc=edu</i> | Domain controllers in the domain that are name servers.   |
| <i>dc=ForestDnsZones,ForestDN</i><br>Example:<br><i>dc=ForestDnsZones,dc=movie,dc=edu</i>       | Domain controllers in the forest that are name servers.   |
| <i>AppPartitionDN</i><br>Example:<br><i>dc=DnsData,dc=movie,dc=edu</i>                          | Domain controllers that have been configured to replicate the custom application partition.   |

Application partitions are treated just like Active Directory domains by DNS. Each application partition has a corresponding DNS domain, which contains records that are used to locate domain controllers that replicate the partition. For the *movie.edu* forest root domain, the *ForestDnsZones* and *DomainDnsZones* default DNS application partitions would translate into *domaindnszones.movie.edu* and *forestdnszones.movie.edu* DNS domains. For the *fx.movie.edu* domain, there would be a *domaindnszones.fx.movie.edu* DNS domain.

## Securing Dynamic Updates

Another huge advantage of storing zones in Active Directory is that you can enable secure dynamic updates. For zones that are not AD-integrated, you have two options

for dynamic updates: allow anyone to make dynamic updates or don't allow dynamic updates at all. Allowing anyone has obvious drawbacks. A malicious client can easily hijack resource records in this mode.

However, when a zone is AD-integrated, you have the option to select **Secure only** for the **Dynamic Updates** configuration, found on the **General** tab for the zone properties in the DNS console. Figure 8-2 shows this window and the three dynamic update options.



Figure 8-2. Dynamic update options

Microsoft uses access control lists (ACLs) on objects in Active Directory to secure zone data and provide secure dynamic update capability. A **Security** tab is available in the DNS console for AD-integrated zones, which allows you to configure whether a user, group, or computer can create and delete objects (i.e., resource records). By default, authenticated computers in a forest can create new records in a zone, and only the client that created a record is allowed to modify it.

## DNS as a Service Location Broker

By now you probably want to hear more about how Active Directory makes use of DNS. At the beginning of the chapter, we dropped the little nugget that when DNS is not available, Active Directory clients may fail to log in. The reason for this is that Active Directory clients use DNS as a *service location broker*, that is, to find the closest server that is providing a certain Active Directory service. Prior to Active Directory,

Windows clients used NetBIOS and WINS to find domain controllers, but hosts running Windows 2000 and later use DNS.

Consider the case of a Windows XP Professional host at Movie U. that's been joined to the *movie.edu* Active Directory domain. When this system boots up, it sends a series of DNS SRV record queries to its configured name server to find the closest domain controller for the *movie.edu* domain.

## The SRV Resource Record

Most of the DNS queries sent by Windows clients during the location process are for SRV (service location) records. The SRV record, introduced in RFC 2052 and updated in RFC 2782, is a general mechanism for locating services. Before we can talk in detail about exactly how a Windows client finds its domain controller using SRV records, we need to describe the SRV record itself.

Locating a service or a particular type of server within a zone is a difficult problem if you don't know which host it runs on. Some administrators have attempted to solve this problem by using service-specific aliases in their zones. For example, at Movie U. we created the alias *ftp.movie.edu* and pointed it to the domain name of the host that runs our FTP archive:

```
ftp.movie.edu.    IN    CNAME    plan9.fx.movie.edu.
```

This makes it easy for people to guess a domain name for our FTP archive and separates the domain name people use to access the archive from the domain name of the host on which it runs. If we were to move the archive to a different host, we could simply change the CNAME record.

Another option, for clients that understand it, is the SRV record. In addition to simply allowing a client to locate the host on which a particular service runs, SRV provides powerful features for load balancing and backup services, similar to what the MX record provides.

A unique aspect of the SRV record is the format of the domain name to which it's attached. Like the service-specific alias *ftp.movie.edu*, the domain name that a SRV record is attached to gives the name of the service sought along with the protocol it runs over, concatenated with a domain name. The labels representing the service name and the protocol begin with an underscore to distinguish them from labels in the domain name of a host. So, for example:

```
_ftp._tcp.movie.edu
```

represents the SRV records someone *ftping* to *movie.edu* should retrieve in order to find the *movie.edu* FTP servers, while:

```
_http._tcp.www.movie.edu
```

represents the SRV records someone accessing the URL *http://www.movie.edu* should look up in order to find the *www.movie.edu* web servers.

The names of the service and protocol must come from the latest Assigned Numbers RFC (the most recent as of this writing is RFC 1700) or be unique names used only locally. Don't use the port or protocol numbers, just the names. When entering SRV records through the DNS console, the service name is limited to eight common services.

The SRV record has four resource record-specific fields: *priority*, *weight*, *port*, and *target*. Priority, weight, and port are unsigned 16-bit numbers (between 0 and 65,535). Target is a domain name.

Priority works similarly to the preference in an MX record: the lower the number in the priority field, the more desirable the associated target. When searching for the hosts offering a given service, clients should try targets with the same priority value before trying those with a higher value in the priority field (lower priority values indicate higher priority—confusing, eh?).

Weight allows zone administrators to distribute load to multiple targets. Clients should query targets of the same priority in proportion to their weight. For example, if one target has a priority of zero and a weight of one and another target has a priority of zero but a weight of two, the second target should receive twice as much load (in queries, connections, etc.) as the first. It's up to the service's clients to direct that load: they typically use a system call to choose a random number. If the number is, say, in the top one-third of the range, they try the first target, and if the number is in the bottom two-thirds of the range, they try the second target.

Port specifies the port on which the service being sought is running. This allows zone administrators to run servers on nonstandard ports. For example, an administrator can use SRV records to point web browsers at a web server running on port 8000 instead of the standard HTTP port (80).

Finally, target specifies the hostname of a host on which the service is running (on the port specified in the port field). Target must be the canonical name of the host (not an alias), with address records attached to it.

So, for the *movie.edu* FTP server, we might add two SRV records to the *movie.edu* zone. Adding the first with the DNS console is shown in Figure 8-3.

After adding the second record, the *movie.edu* zone datafile (*movie.edu.dns*) contains these records:

```
_ftp._tcp.movie.edu. IN SRV 1 0 21 plan9.fx.movie.edu.  
                     IN SRV 2 0 21 thing.fx.movie.edu.
```

This instructs SRV-capable FTP clients to try the FTP server on *plan9.fx.movie.edu*'s port 21 first when accessing *movie.edu*'s FTP service and then to try the FTP server on *thing.fx.movie.edu*'s port 21 if *plan9.fx.movie.edu*'s FTP server isn't available.

The records:

```
_http._tcp.www.movie.edu. IN SRV 0 2 80 www.movie.edu.  
                          IN SRV 0 1 80 www2.movie.edu.  
                          IN SRV 1 1 8000 postmanrings2x.movie.edu.
```

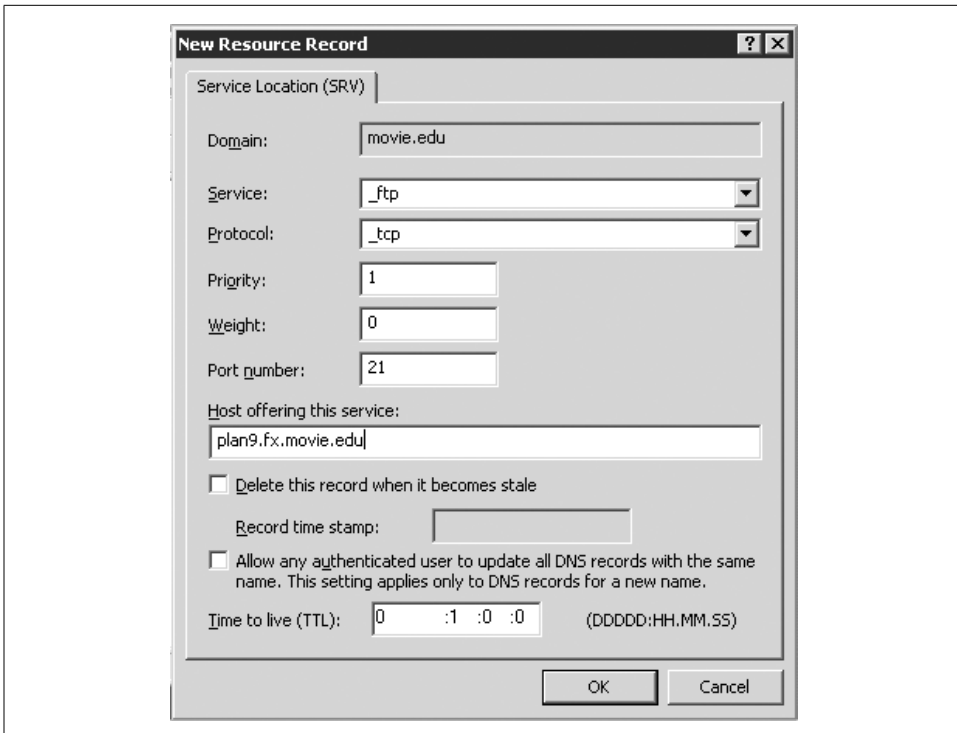


Figure 8-3. Adding an SRV record with the DNS console

direct web queries for *www.movie.edu* (the web site) to port 80 on *www.movie.edu* (the host) and *www2.movie.edu*, with *www.movie.edu* getting twice the queries *www2.movie.edu* does. If neither is available, queries go to *postmanrings2x.movie.edu* on port 8000.

But don't get excited and add SRV records for your FTP and web servers: currently few clients actually use SRV records to locate their servers. In fact, we're not aware of any FTP clients or web browsers that look up SRV records. On the other hand, when Microsoft was looking for a way to have Windows-based clients find their domain controllers, SRV records fit the bill perfectly.

## DC Locator

One of the fundamental issues for clients in any networked environment is finding the optimal server to authenticate against. The process under Windows NT was not very efficient and could cause clients to authenticate to domain controllers in the least optimal location. With Active Directory, clients use DNS to locate domain controllers via the DC locator process. To illustrate at a high level how the DC locator process works, here's an example where a client has moved from one location to another and needs to find a domain controller (DC).

1. A client previously located in Site A logs in from Site B.
2. When the client boots up, it thinks it is still in Site A, so it proceeds to contact the DC it has cached locally in the registry.
3. The DC in Site A receives the request and realizes that the client should now be talking to a DC in Site B, since its IP address has changed. In its reply to the client, the DC in Site A refers the client to the DC in Site B.
4. The client then performs a DNS lookup to find a DC in Site B.
5. The client then contacts a DC in Site B. Three things can happen: the DC responds and authenticates the client; the DC fails to respond (it could be down), so the client attempts to use a different DC in Site B; or the DC fails to respond and the client fails to find another DC in Site B. In the last case, the client turns back to the DC in Site A and authenticates with the first server it contacted.

Two things are needed to support the DC locator process: the site topology must be properly defined in Active Directory, and DNS must contain the necessary Active Directory SRV records, which we describe in the next section. For a more detailed description of how the DC locator process works, including the specific resource records that are used during the process, check out Microsoft Knowledge Base articles 247811 and 314861.

## Resource Records Used by Active Directory

When you promote a domain controller into a domain, the file `%SystemRoot%\System32\Config\netlogon.dns` is generated. This file contains the necessary resource records for the DC to function correctly within Active Directory. The NetLogon service keeps this file updated based on site membership, GC status, and site coverage.

The contents of the file looks like the following for a DC named `terminator.movie.edu` in the `movie.edu` domain with the IP address 10.1.1.1. We've reordered the file a bit to group records of similar purpose together. Note that some lines may wrap due to their length.

```
movie.edu. 600 IN A 10.1.1.1
ec4caf62-31b2-4773-bcce-7b1e31c04d25._msdcs.movie.edu. 600 IN CNAME↵
terminator.movie.edu.
gc._msdcs.movie.edu. 600 IN A 10.1.1.1
_gc._tcp.movie.edu. 600 IN SRV 0 100 3268 terminator.movie.edu.
_gc._tcp.Default-First-Site-Name._sites.movie.edu. 600 IN SRV 0 100 3268↵
terminator.movie.edu.
_ldap._tcp.gc._msdcs.movie.edu. 600 IN SRV 0 100 3268 terminator.movie.edu.
_ldap._tcp.Default-First-Site-Name._sites.gc._msdcs.movie.edu. 600 IN SRV 0 100 3268↵
terminator.movie.edu.
_kerberos._tcp.dc._msdcs.movie.edu. 600 IN SRV 0 100 88 terminator.movie.edu.
_kerberos._tcp.Default-First-Site-Name._sites.dc._msdcs.movie.edu. 600 IN SRV 0 100↵
88 terminator.movie.edu.
_kerberos._tcp.movie.edu. 600 IN SRV 0 100 88 terminator.movie.edu.
```

```

_kerberos._tcp.Default-First-Site-Name._sites.movie.edu. 600 IN SRV 0 100 88.
terminator.movie.edu.
_kerberos._udp.movie.edu. 600 IN SRV 0 100 88 terminator.movie.edu.
_kpasswd._tcp.movie.edu. 600 IN SRV 0 100 464 terminator.movie.edu.
_kpasswd._udp.movie.edu. 600 IN SRV 0 100 464 terminator.movie.edu.
_ldap._tcp.movie.edu. 600 IN SRV 0 100 389 terminator.movie.edu.
_ldap._tcp.Default-First-Site-Name._sites.movie.edu. 600 IN SRV 0 100 389.
terminator.movie.edu.
_ldap._tcp.pdc._msdcs.movie.edu. 600 IN SRV 0 100 389 terminator.movie.edu.
_ldap._tcp.97526bc9-adf7-4ec8-a096-0dbb34a17052.domains._msdcs.movie.edu. 600 IN SRV.
0 100 389 terminator.movie.edu.
_ldap._tcp.dc._msdcs.movie.edu. 600 IN SRV 0 100 389 terminator.movie.edu.
_ldap._tcp.Default-First-Site-Name._sites.dc._msdcs.movie.edu. 600 IN SRV 0 100 389.
terminator.movie.edu.
DomainDnsZones.movie.edu. 600 IN A 10.1.1.1
_ldap._tcp.DomainDnsZones.movie.edu. 600 IN SRV 0 100 389 terminator.movie.edu.
_ldap._tcp.Default-First-Site-Name._sites.DomainDnsZones.movie.edu. 600 IN SRV 0 100.
389 terminator.movie.edu.

```

Let's go through what these records are actually used for, splitting them up into sections for ease of understanding. To start with, the first record is for the domain itself:

```
movie.edu. 600 IN A 10.1.1.1
```

This A record is for LDAP clients that don't understand SRV records. Since Windows clients use SRV records to locate the LDAP service on the domain controller, you don't need this A record (the one for *movie.edu*) unless you're running other LDAP clients. (And even then, you can just point those clients at the domain controller using its fully qualified name: *terminator.movie.edu*, in this case.) It's good that the A record isn't required because a lot of folks already have an A record at the apex of their DNS namespace. This record usually points to a web server, not to an Active Directory server. For example, Movie U.'s main web server is accessible via both *www.movie.edu* and *movie.edu*.

Next, we have the following record:

```
ec4caf62-31b2-4773-bcce-7b1e31c04d25._msdcs.movie.edu. 600 IN CNAME.
terminator.movie.edu.
```

This CNAME record, found under the *\_msdcs* subdomain, is used by domain controllers during replication. The left side of the record is the globally unique ID (GUID) for the server. DCs use this record if they know the server's GUID and want to determine its hostname. If this record isn't present for a DC, other DCs cannot replicate changes from it. The *DNSLint* command, available in the Windows Server 2003 Support Tools, can be used to make sure all DCs have this CNAME record present. Here is an example command line:

```
C:\> dnslint /ad 10.13.51.18 /s 10.13.52.15 /v
```

The required */ad* switch is optionally followed by the IP address of a DC to perform LDAP queries against. If an IP address is not specified, the local host is assumed. The */s* option, also required, must be followed by the IP address (or the text *localhost*) of

a DNS server that is authoritative for the *\_msdcs* subdomain of the DC's domain. See Chapter 15 for more on using *DNSLint* for troubleshooting.

Next, we have this A record:

```
gc._msdcs.movie.edu. 600 IN A 10.1.1.1
```

This record is registered only if the DC is a global catalog server. You can query *gc.\_msdcs.movie.edu* to obtain a list of all the global catalog servers in the forest in much the same way you could query the AD domain's A record to get a list of all the domain controllers for a domain (if you allow registration of that record by domain controllers).

A few more global catalog–specific records are shown next:

```
_gc._tcp.movie.edu. 600 IN SRV 0 100 3268 terminator.movie.edu.  
_gc._tcp.Default-First-Site-Name._sites.movie.edu. 600 IN SRV 0 100 3268.┘  
terminator.movie.edu.  
_ldap._tcp.gc._msdcs.movie.edu. 600 IN SRV 0 100 3268 terminator.movie.edu.  
_ldap._tcp.Default-First-Site-Name._sites.gc._msdcs.movie.edu. 600 IN SRV 0 100 3268.┘  
terminator.movie.edu.
```

Note that in the case of global catalog SRV records, the fourth record-specific data field—which is used for the port for the service—is 3268, the global catalog port. You may have also noticed the entries that contain **Default-First-Site-Name**. Each global catalog server registers site-specific records so clients can find the optimal global catalog server based on their site membership. If you renamed the default site and have other sites defined, you will see these site names used in the SRV records. See the “Site Coverage” sidebar for more information.

The next few SRV records are for Kerberos authentication (port 88) and the Kpasswd process (port 464), which allows users to change passwords:

```
_kerberos._tcp.dc._msdcs.movie.edu. 600 IN SRV 0 100 88 terminator.movie.edu.  
_kerberos._tcp.Default-First-Site-Name._sites.dc._msdcs.movie.edu. 600 IN SRV 0 100.┘  
88 terminator.movie.edu.  
_kerberos._tcp.movie.edu. 600 IN SRV 0 100 88 terminator.movie.edu.  
_kerberos._tcp.Default-First-Site-Name._sites.movie.edu. 600 IN SRV 0 100 88.┘  
terminator.movie.edu.  
_kerberos._udp.movie.edu. 600 IN SRV 0 100 88 terminator.movie.edu.  
_kpasswd._tcp.movie.edu. 600 IN SRV 0 100 464 terminator.movie.edu.  
_kpasswd._udp.movie.edu. 600 IN SRV 0 100 464 terminator.movie.edu.
```

Just as with the global catalog SRV records, there may be more of the site-specific Kerberos records for any additional sites the DC covers.

The rest of the SRV records are used to represent a domain controller for a particular domain, site, and application partition. One record to note is the *\_ldap.\_tcp.pdc.\_msdcs.movie.edu* entry, which is registered by the DC that is acting as the primary domain controller (PDC) emulator for the domain.

```
_ldap._tcp.movie.edu. 600 IN SRV 0 100 389 terminator.movie.edu.  
_ldap._tcp.Default-First-Site-Name._sites.movie.edu. 600 IN SRV 0 100 389.┘  
terminator.movie.edu.
```

## Site Coverage

You can create sites in the Active Directory topology that do not have domain controllers located in that site (i.e., DC-less sites). In this situation, the domain controllers that have the lowest cost as defined by the site links *cover* for that site. When a DC covers for a site, it adds site-specific SRV records so that it advertises itself as a DC that can handle queries for clients in the site. To see a list of the sites that a particular DC is covering for, run the following *nltest* command (contained in the Windows Support Tools) and replace *dc01* with the name of the DC you want to query:

```
C:\> nltest /dsgetsitecov /server:dc01
```

So why would you want DC-less sites, especially when you could just add the subnets in those sites to the closest site that contains a domain controller? As far as Active Directory goes, there aren't many benefits to creating DC-less sites. One possible benefit is that it allows you to mimic your physical network topology in your site topology. In that situation, it is very likely that you have sites that do not have domain controllers. The real benefits come in with other services that depend on the site topology, such as DFS. It is possible that you might need a DFS server in a remote site that does not contain a domain controller. If you created the DC-less site, the clients in that site would use the local DFS server instead of one in the next closest site.

Another benefit of DC-less sites is being able to apply group policy at the site level. If you have small sites that need a distinct group policy, you can create a small site group policy object and apply it to the target sites.

```
_ldap._tcp.pdc._msdcs.movie.edu. 600 IN SRV 0 100 389 terminator.movie.edu.  
_ldap._tcp.97526bc9-adf7-4ec8-a096-0dbb34a17052.domains._msdcs.movie.edu. 600 IN SRV 0  
0 100 389 terminator.movie.edu.  
_ldap._tcp.dc._msdcs.movie.edu. 600 IN SRV 0 100 389 terminator.movie.edu.  
_ldap._tcp.Default-First-Site-Name._sites.dc._msdcs.movie.edu. 600 IN SRV 0 100 389  
terminator.movie.edu.  
DomainDnsZones.movie.edu. 600 IN A 10.1.1.1  
_ldap._tcp.DomainDnsZones.movie.edu. 600 IN SRV 0 100 389 terminator.movie.edu.  
_ldap._tcp.Default-First-Site-Name._sites.DomainDnsZones.movie.edu. 600 IN SRV 0 100  
389 terminator.movie.edu.
```

Note the last three records—they are for the *DomainDnsZones* application partition. As we mentioned early, the DC locator process is used to find domain controllers for domains, but it is also used for application partitions as well. There will be records like those three for all the application partitions *terminator* replicates.

Based on all of these records, you can obtain a lot of information about an Active Directory environment by doing simple DNS queries. Some of the information you can retrieve includes:

- Global catalog servers in a forest or a particular site
- Kerberos servers in a domain or a particular site

- Domain controllers in a domain or a particular site
- PDC emulator for a domain

We could go on and on about Active Directory, but this is a book about DNS, so we won't. If you are interested in more information on Active Directory, see *Active Directory*, Second Edition and *Active Directory Cookbook*, both from O'Reilly. We return to a core DNS issue in the next chapter: growing your domain.