Chapter 6

DNS and Name Resolution in Windows Server 2012 R2

Computers communicate with each other using IP addresses, whether IPv4 or IPv6. However, it's difficult for most people to remember the IP address for their favorite website or file server. They like using friendly, text-based names. Thus, naming systems are implemented to resolve the friendly names of computers with their assigned IP addresses. The Domain Name System (DNS) is the naming system that Windows Server 2012 R2 servers use. Not only does DNS help users to easily identify devices, but many other services such as Active Directory require DNS so that clients and servers can locate and communicate with domain controllers.

In this chapter, you will learn to:

- Explain the fundamental components and processes of DNS
- Configure DNS to support an Active Directory environment
- Manage and troubleshoot DNS resolution for both internal and external names

Understanding the DNS Server Role

DNS has been around for decades prior to Microsoft developing its first edition of DNS in Windows NT 4.0. There are many different varieties of DNS implementations that support the required features and processes that define DNS. In this section, we'll cover the concepts of Microsoft's Domain Name System and how it's applied in Windows operating systems.

DNS is implemented within Windows Server 2012 R2 to manage IP address name resolution. Once installed on a server, the DNS service will need to communicate with other DNS name servers, which is accomplished using several different methods, such as forwarding, root hints, and delegation. The DNS service will also maintain databases, named *zones*, for the internal Active Directory domain or other namespaces. The domain computers will need to query this DNS service, so you must configure individual computers in order to provide efficient and rapid name resolution.

Windows Server 2012 R2 and previous releases of the Windows Server operating system offer a built-in DNS Server role. Windows Server 2012 R2 DNS is compatible with older versions of DNS going back to Windows Server 2003; however, versions earlier than Windows Server 2008 do not support IPv6 and are functional only with IPv4 addresses.

The following is a short summary of the DNS fundamental concepts referred to in this chapter:

Hostname This is the (friendly) name of a computer. According to DNS standards, it can be up to 255 characters in length. It is equivalent to a computer's first name, for example, EC01.

Namespace This is the name of a domain, not specifically an Active Directory domain though. It's a logical set of hosts signified by a name controlled by a set of name servers. This is equivalent to a computer's last name; they're all part of the same family. For example, Bigfirm .com is the namespace for hosts in the Bigfirm.com domain.

Fully Qualified Domain Name The FQDN is the hostname appended to the domain's namespace, such as EC01.Bigfirm.com.

HOSTS File This is a text file that statically maps hostnames to IP addresses. This file is located in `c:\windows\system32\drivers\etc` for standard Windows Server 2012 R2 installations and can be used as an entry-level alternative to a DNS server for name resolution in small environments. Don't try to manage a large enterprise environment with static HOSTS file entries though, because it will quickly turn into an administrative nightmare!

Name Server This is a DNS server that will resolve FQDNs to IP addresses. Name servers also control namespaces for specified domains. They will resolve requests for that namespace from DNS clients throughout the network.

Hierarchical Naming Structure The namespace is created so that the left part of a name is a subset of the right part of the name as shown in the FQDN. With this, the naming servers can start at the right side of the name, and the responses from the name servers will direct it to the correct naming server for a given namespace. For example, as shown in Figure 6.1, Ec01.Ecoast.Bigfirm.com is the FQDN for a server in the Ecoast.Bigfirm.com domain. This domain is actually a subset, or a *subdomain*, under the control of the Bigfirm.com domain. You can say the same thing about Bigfirm.com for the .com top-level domain name. The strength is that you can ask the .com domain name server where the Bigfirm.com name server is. You can do the same for the Ecoast.Bigfirm.com server, and so on. The FQDN name directs the query to the right name server through a process named *recursion*.

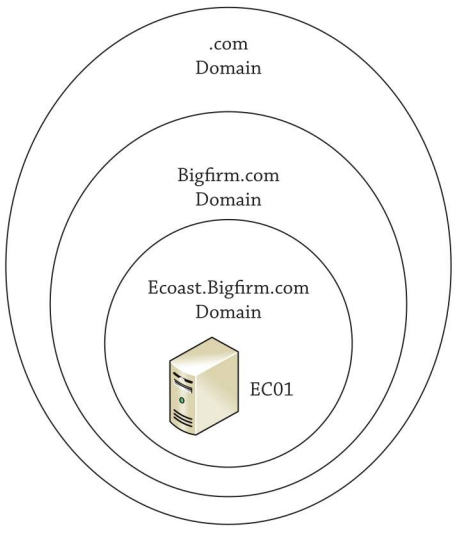


FIGURE 6.1 Hierarchical DNS naming

Recursion This is a server-directed process to resolve an FQDN. If the server cannot resolve the FQDN with its own information, it will send the query to other name servers. The recursion process comprises root servers and domain name servers. Root servers are the top of the hierarchical naming structure. The root servers list the name servers that control the top-level domain names such as .com, .gov, and .edu. The top-level domain servers control the registry of subdomains beneath the top-level domain. For example, name servers for the Sybex.com subdomain are registered on the .com domain servers. When a query occurs, the following occurs (as shown in Figure 6.2):

1. The DNS client requests a name, like www.Sybex.com, from its DNS server.
2. Through the recursive process, the DNS server queries the root servers for the .com domain name servers.
3. The root servers give a list of name servers for the .com domain.
4. Then the DNS server queries the .com name servers for Sybex.com.
5. It receives another list of name servers for the Sybex.com domain.
6. It queries the provided name servers for the www.Sybex.com FQDN.
7. The Sybex.com DNS server coughs up the IP address of the www server to the DNS server.
8. The DNS server passes the IP address to the client.
9. Armed with the IP address, the client connects with the web server www.Sybex.com.

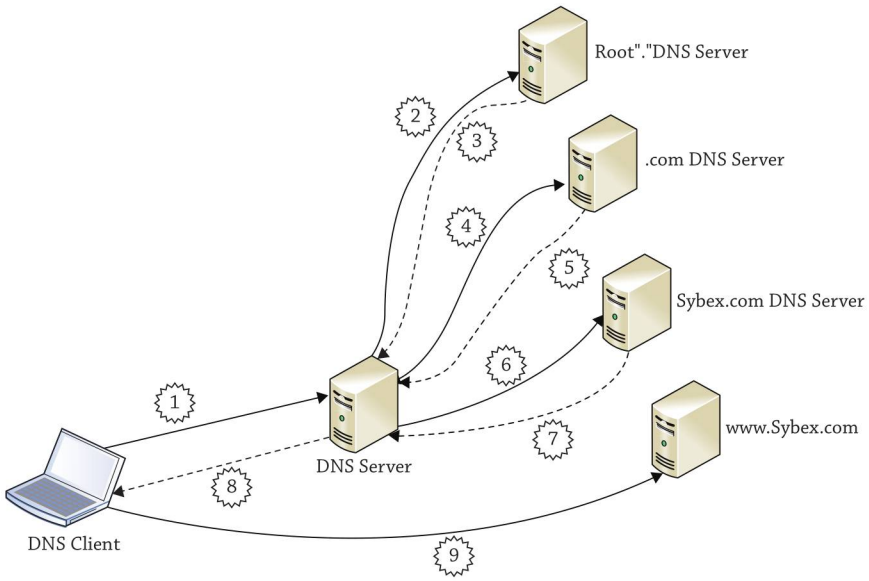


FIGURE 6.2 DNS recursion process

Delegation This means allowing another name server to control a subdomain of a given namespace. For example, the Bigfirm.com name servers can delegate control of the Ecoast .Bigfirm.com namespace to another server.

Forwarding This is an alternative to the recursion process. Forwarding is a lateral request to another name server within the network. The forwarding server obtains a response and relays it to the originating name server.

Iteration This is a client-directed process to resolve an FQDN. If the client receives a negative request from a name server, it will query another name server.

NetBIOS Naming System This legacy naming system was used primarily within old Microsoft NT 4.0 networks. Its processes are still part of modern-day Windows operating systems, however, particularly when using non-domain (workgroup)–based computers.

Service Records Service records (SRVs) are records within a DNS namespace to resolve a service to a hostname. This is an essential part of DNS supporting Active Directory.

Dynamic DNS Update Dynamic DNS (DDNS) update is a process that allows DNS clients to register their hostnames in an assigned namespace such as DHCP. This reduces the need of admins to manually enter records in the name server databases. This is another essential part of DNS supporting Active Directory.

Installing DNS

DNS for Windows Server 2012 R2 can be deployed into several different configurations depending on your scenario. You can choose to install a stand-alone DNS server onto a non-domain-joined computer, or you can deploy it onto either your domain-joined member servers or Active Directory domain controller servers. Whatever the scenario, installing the DNS role is simple. This section will first explain what you need to do to manually deploy DNS onto a non-domain-joined computer in a stand-alone configuration. Later, we will explain how to automatically configure DNS to integrate with Active Directory so as to ensure name resolution runs smoothly inside your domain environment.

Configuring a Stand-Alone DNS Server

First things first: you'll need to have allocated a static IP address to the server that you want to install the DNS role onto, since hitting a moving target is pretty tough for the DNS client! From the desktop, if you hit Ctrl+R and then type **`ncpa.cpl`** and hit Enter, you will be brought to the Network Connections window. Here, you can right-click your network adapter and then select Properties to open the configuration window. If you double-click Internet Protocol Version 4 (TCP/IPv4), you will be presented with the TCP/IPv4 Properties dialog box. Once there, you can then assign a static IP address, as shown in Figure 6.3.

When you have assigned a static IP address, you should then add the primary DNS suffix, such as Bigfirm.com, into the Advanced TCP/IPv4 Settings dialog box, as shown in Figure 6.4.

We use the word *should* here because it isn't always necessary to add a primary DNS suffix. The primary DNS suffix is modified automatically when the computer joins a domain. If the server is going to operate as part of a workgroup (as is the case in our example here), you will need to add it so other DNS servers can locate it within the DNS structure and the DNS service is properly configured during installation.

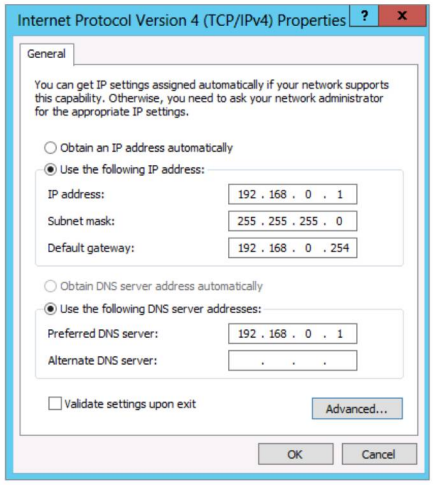


FIGURE 6.3
Assigning a static IP address

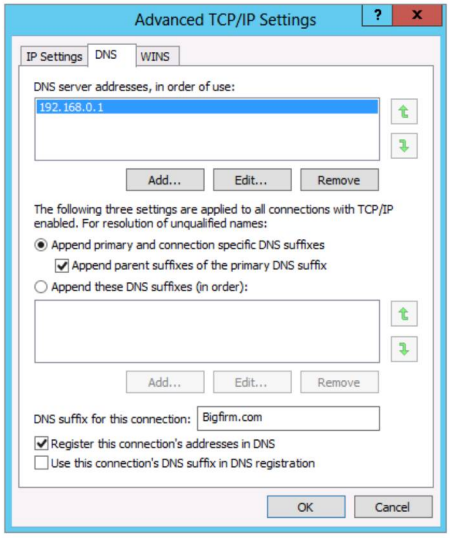


FIGURE 6.4
Adding the DNS suffix

Once you've configured a static IP address and DNS suffix on your non-domain-joined server, follow these steps to install the DNS role:

1. Open Server Manager, click Dashboard, and then select the "Add roles and features" option, as shown in Figure 6.5.
2. At the "Before you Begin" dialog box inside the Add Roles and Features Wizard, click Next to continue.
3. Choose the "Role-based or Feature-based Installation" option from the Select Installation Type dialog box; then click Next.

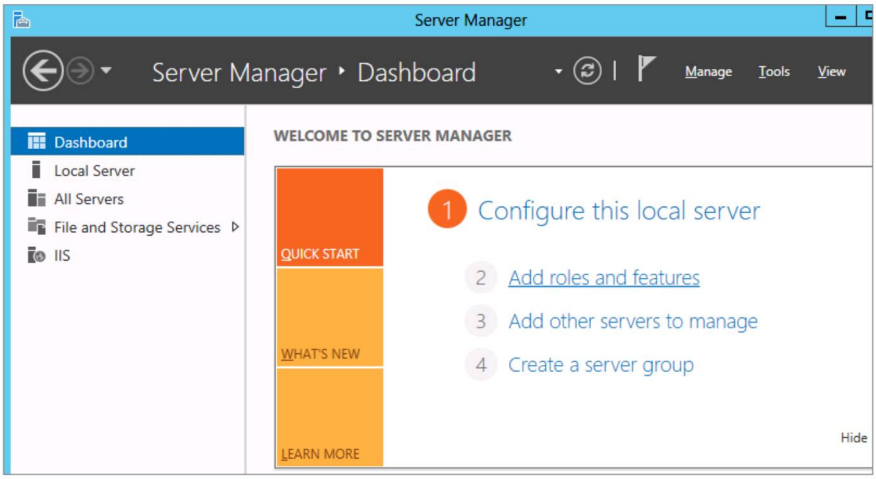


FIGURE 6.5
Adding the DNS role

4. Click the "Select a Server from the Server Pool" option and ensure that your server is highlighted in the Select Destination Server dialog box; then click Next to move on.
5. In Select Server Roles dialog box, choose the DNS Server option from the list (if applicable, click the Add Features button from the Add Roles and Features Wizard dialog box that pops up after you select the DNS Server role), as shown in Figure 6.6.

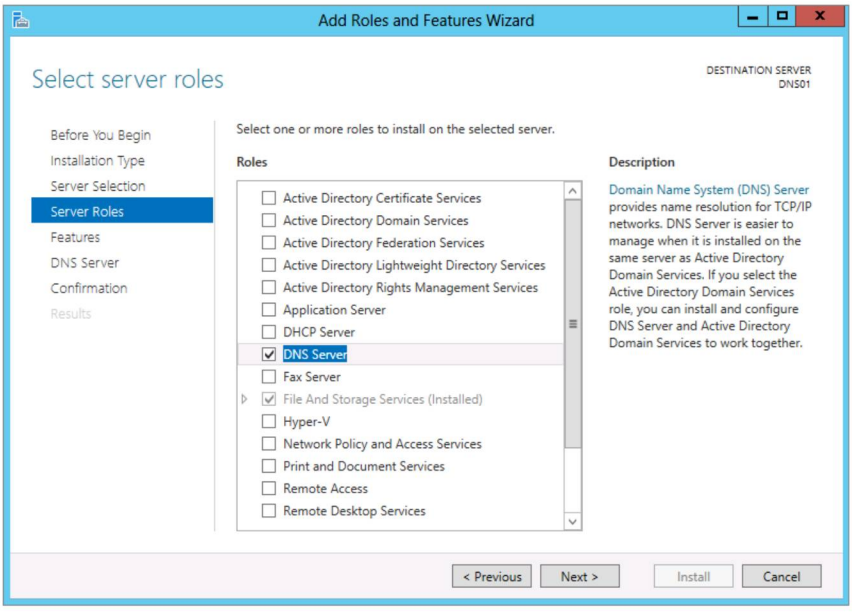


FIGURE 6.6
Selecting the DNS Server role

6. Click the Next button in the remaining dialog boxes until you reach the Confirm Installation Selections dialog. Verify that all of your selections are correct, and then click Install to begin the installation.

7. When the DNS Server role installation has completed, click the Close button to finish the process.

Installing the role in this way creates a solitary DNS name server that is talking only with the Internet root servers. It can support a LAN environment for resolving Internet names, but that's about it. The Domain Name System leverages other naming servers to resolve names throughout the DNS structure. For this reason, you'll have to configure the server to talk with other DNS servers that exist in the internal network. Although configuring your DNS server in a stand-alone non-domain configuration can have its merits (such as deploying one in environments where managing multiple static HOSTS files across servers becomes a pain), you will really see the power of Windows Server 2012 R2 DNS when you use it with Active Directory.

Integrating with Other DNS Servers

In "Understanding the DNS Server Role," we mentioned that there are different methods for resolving DNS names, such as forwarding, recursion, delegation, and iteration. These methods are related to the integration with other DNS servers. Before we get started, remember that iteration is basically client driven. If the DNS server doesn't have an answer, the client will go to another DNS server. The server or the client can be configured for iteration only, but it is not the default and is rarely implemented. The other three—forwarding, recursion, and delegation—involve contacting other DNS servers by the queried DNS server.

Recursion is the primary process occurring on the Internet. The queried DNS server starts at the top and works its way down with the referrals it receives from each DNS server it contacts. In Windows DNS servers, the top servers are listed on the Root Hints tab of the DNS server properties, as shown in Figure 6.7. You can display this in the DNS Management snap-in by right-clicking the server icon and selecting Properties. By default, it is populated with "live" Internet DNS servers.

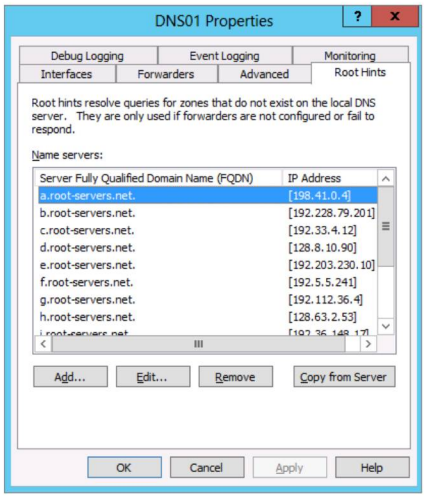


FIGURE 6.7
Root Hints tab

The list is located in a text file named Cache in the `c:\windows\system32\dns` folder displayed in Figure 6.8.

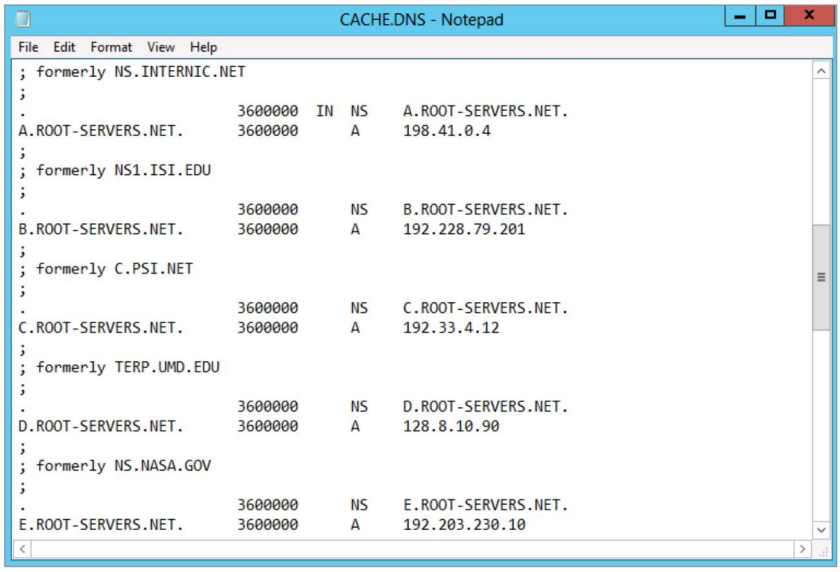


FIGURE 6.8
The Cache file listing root hints

In a single Active Directory domain environment, you can leave this alone. DNS servers can use these references to resolve Internet-based namespaces such as Sybex.com when a client requests it. In larger environments, you can remove the root hint entries on other DNS servers and rely on one server to support DNS resolution throughout the external environment. This would essentially be the caching DNS server of the internal structure.

While root hints manage the queries going up the DNS hierarchical structure, delegation manages queries going downward. In our example, the DNS servers that control the .com namespace delegate the control of registered subdomains like Sybex.com. The delegation is simply the listing of these servers. So, the .com name server sends the list of name servers to a DNS server looking for the Sybex.com namespace.

In a Windows environment, you can see delegation in play with multiple Active Directory domains. If you have an Active Directory domain named Bigfirm.com, you have an associated Bigfirm.com DNS namespace. You could create an Active Directory domain named Ecoast.Bigfirm.com. Instead of keeping all the DNS namespaces on the Bigfirm.com DNS server, you can delegate the Ecoast.Bigfirm.com DNS namespace to another DNS server.

Figure 6.9 illustrates this delegation in the DNS management console. The DC01 DNS server supports a forward lookup zone named Bigfirm.com. The Ecoast subdomain, represented by an icon of a gray folder with a text file on top of it, lists only a name server record for EC1.Ecoast.Bigfirm.com with its IP address.

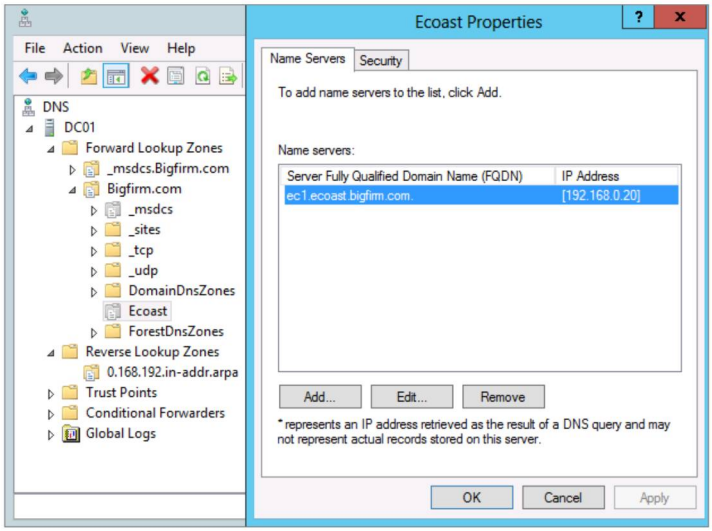


FIGURE 6.9
Delegated domain for Ecoast.Bigfirm.com

A forwarder is another DNS server to request a lateral query. When a server cannot resolve the DNS name, it can forward the request to another DNS server rather than going through the root hints. In an internal DNS environment, forwarders can be used to resolve other namespaces. For example, the DNS server EC1.Ecoast.Bigfirm.com needs to resolve Bigfirm.com servers and others namespaces, so a forwarder is entered in its properties, as shown in Figure 6.10.

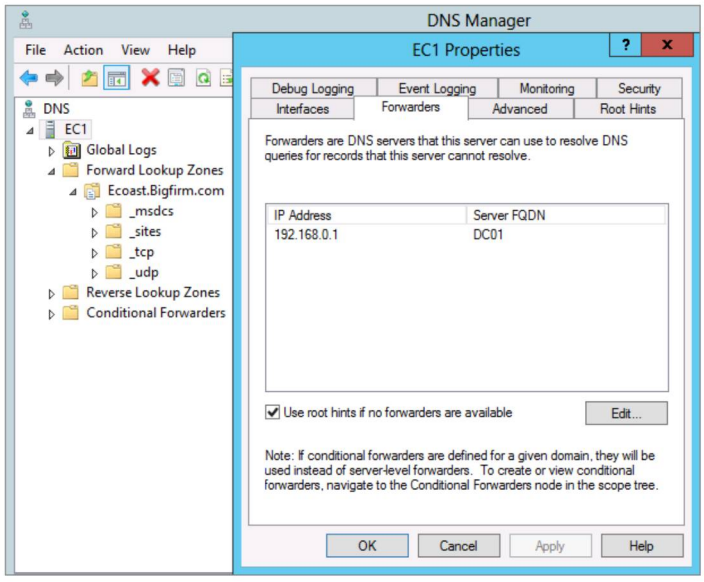


FIGURE 6.10
Forwarders tab

In Figure 6.10, notice the check box "Use root hints if no forwarders are available." You can disable this in a larger environment if you want to centralize the Internet-based DNS queries. Also notice the text concerning conditional forwarders.

Conditional forwarders have their own node in the DNS Manager's scope tree in the left pane. To manage resolution of a specific namespace, the conditional forwarder can direct queries to a specific server. In Figure 6.11, a conditional forwarder was set up for the Otherdomain .local namespace. Thus, any queries for this namespace will go to OS01.Otherdomain.local DNS server.

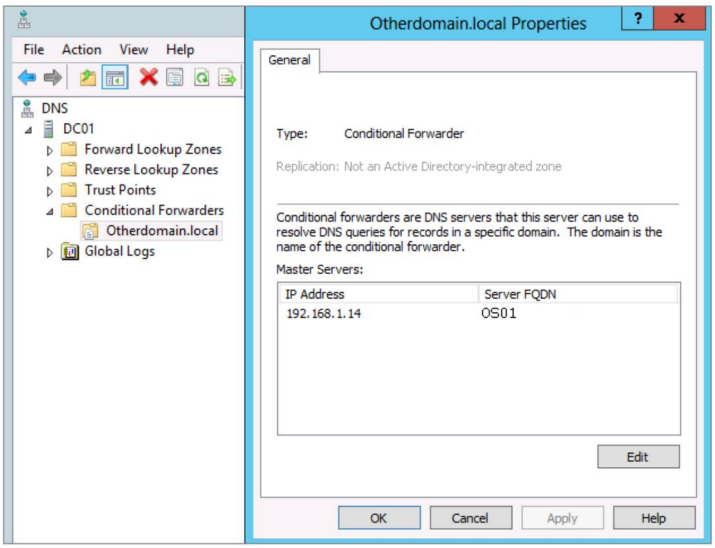


FIGURE 6.11 Conditional forwarder

Conditional forwarders are created by right-clicking the Conditional Forwarder folder in the DNS management console and selecting New Conditional Forwarder. The New Conditional Forwarder dialog box provides an option to replicate the setting to other domain controllers in the domain or forest using Active Directory application partitions.

Forwarding also can be used to resolve Internet-based queries instead of using root hints. We prefer to use this in small environments that have an Internet service provider such as cable or DSL. These ISPs will have their own DNS servers that are found on the router configuration. So, these are entered as forwarders on the internal DNS servers. Although root hints can work in this environment, we find this technique more reliable. In addition, it limits the internal DNS server's communication to a specified external source.

The integration with other DNS servers can complete the DNS role configurations. The DNS server could receive requests and then send these requests to other DNS servers. Once it receives the answer, it could cache the information for a period of time, which in Windows Server 2012 R2 is set to one hour by default. This configuration is referred to as a *caching-only server*. If the server is to control a namespace, you have to add zones.

Implementing Zones to Manage Namespaces

A *zone* is the database for a namespace. On the Internet, there is a DNS server that controls the Sybex.com namespace. If you want the IP address for www.sybex.com, this DNS server will look in its zone (database) to find the answer. So, you can create zones on DNS servers to manage namespaces.

In Windows DNS servers, there are four types of zones:

- Standard primary
- Standard secondary
- Active Directory integrated
- Stub

The stub zone doesn't manage a namespace, however, and is more like a conditional forwarder. We'll discuss each of these zone types in the following sections.

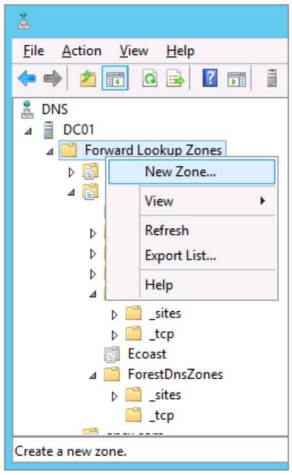
Understanding the Standard Primary Zone

Name servers were designed to centralize name resolution for a network. Initially, a DNS server responded to requests based on its text-based HOSTS file. This is essentially what Microsoft named a *standard primary zone*. The standard primary zone is a text file in which the server maintains the records for a given namespace. That's what is *standard* about the Windows DNS implementation. *Primary* refers to replication.

Way back in the old days of Windows NT, there was one master domain controller named the *primary domain controller* (PDC), which controlled any writing to its database. The rest were *backup domain controllers* (BDCs), which had read-only copies. In DNS terms, primary zones mean there is only one master, and this server is it. Other DNS servers can have only read-only copies of this zone; these are secondary.

Creating a zone is easily accomplished with the New Zone Wizard, which you can initiate inside DNS Manager by right-clicking Forward Lookup Zones and selecting New Zone, as shown in Figure 6.12.

Figure 6.12
Creating a new zone



The New Zone Wizard prompts for the following information:

- The namespace or name of the domain such as Primaryzone.local.
- The name of the text file, which defaults to the `.dns` file extension.
- The Dynamic DNS Update option. We'll discuss this later in the "Updating DNS Dynamically" section.

After the zone creation, you can view the contents of the text file in the `c:\windows\system32\dns` folder displayed in Figure 6.13. Additional records, `CNAME` and `A`, were created for examples that are located at the bottom of the file.

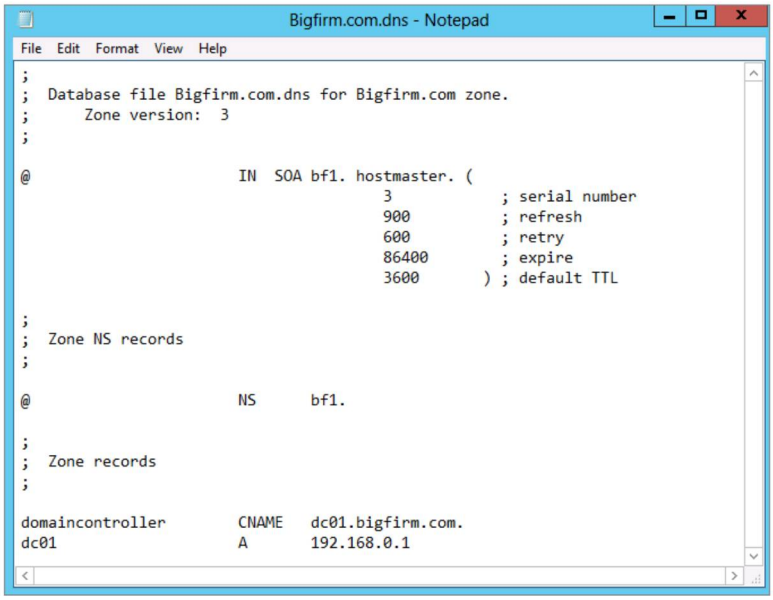


Figure 6.13 Standard primary zone file

Understanding the Standard Secondary Zone

The *standard secondary zone* is the read-only copy of the standard primary zone or an Active Directory integrated zone. Replication is performed through the zone transfer process, which is configured on the zone's properties. On Windows DNS servers, the default setting for zone transfers is to allow transfers to only the registered name servers of the zone, as shown in Figure 6.14.

You can add a name server, such as EC1.Ecoast.Bigfirm.com to the Name Servers tab, as in Figure 6.15, to permit replication to this server.

Once that is accomplished, you can run through the New Zone Wizard to create a standard secondary zone on EC1. This will require the master server's IP address, which would be the server to request the zone transfer. It doesn't necessarily have to be the DNS server with the standard primary zone. The result is a successful transfer of the zone to EC1, as shown in Figure 6.16.

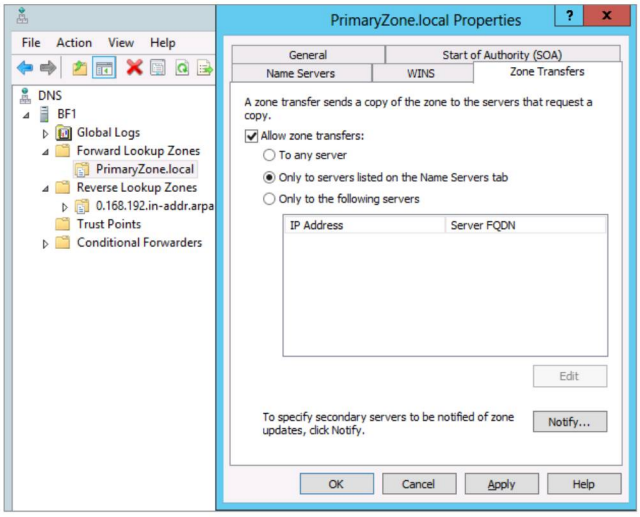


FIGURE 6.14
Zone Transfers tab for a primary zone

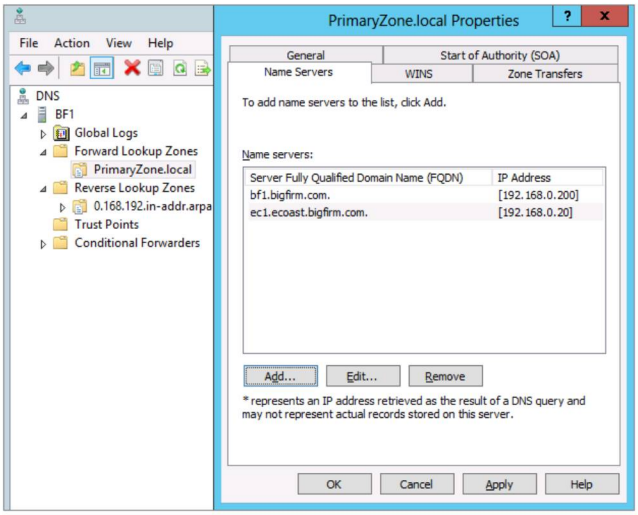


FIGURE 6.15
Name Servers tab for a primary zone

The zone transfer process is not complex. The server for the primary zone keeps track of the changes it has made and has a serial number for the change. When a secondary server contacts the primary server, it checks out the serial number in the Start of Authority record. If the serial number on the secondary server doesn't match, it's time to replicate the changes. This is simply a text-based blast of the database information. Earlier versions of DNS supported AXFR (all zone transfers) replication, which meant the entire zone was replicated to the secondary server. This could be too much traffic to throw onto the line. Windows DNS supports IXFR (which are incremental zone transfers), which replicates just the changes. Windows DNS also supports notification of secondary servers, which reduces the wait time to trigger replication.

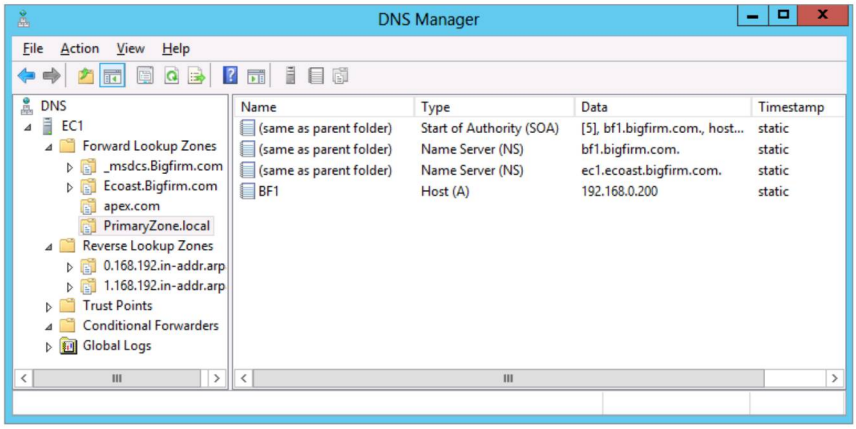


FIGURE 6.16 Standard secondary zone

UNDERSTANDING ACTIVE DIRECTORY INTEGRATED ZONES

The third zone, Active Directory Integrated, is the predominant implementation of Windows DNS servers. The name Active Directory says it all:

- First, the DNS records are stored in the Active Directory database rather than a text file.
- Second, the zones are replicated to all other Active Directory domain controllers in the domain rather than through the zone transfer process.

Since the Active Directory database uses multimaster replication, changes can be made to the DNS zone on any domain controller, and they would be replicated to the other domain controllers. With the integration of DNS in Active Directory, the coupling of DNS and domain controller roles became the norm. For more information concerning the replication process, refer to Chapter 22.

Like the standard zones, an Active Directory integrated zone can be created with the New Zone Wizard. On the first page of the wizard, shown in Figure 6.17, you select the "Store the zone in Active Directory" check box.

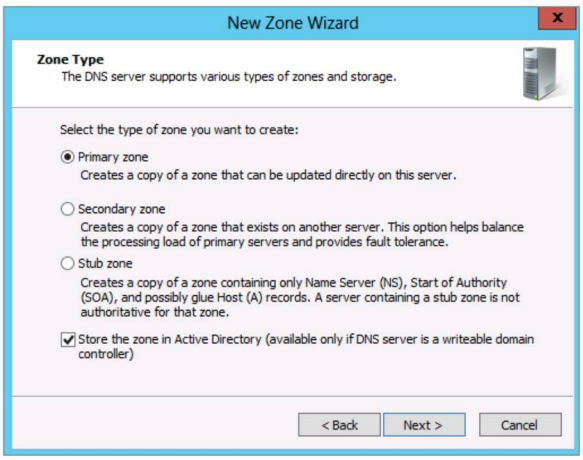


FIGURE 6.17 Creating an Active Directory integrated zone

On the next page of the New Zone Wizard, shown in Figure 6.18, you have four options: forest wide, domain wide, domain wide (Windows 2000–compatible), and a specified custom application partition for storing the zone database (this last option is typically grayed out by default, but we will explain in the next paragraph how to enable this). The first two options place the database in automatically created default application partitions: one for the forest and one for the domain that the domain controller is a member of. The Windows 2000–compatible location is the domain partition of the Active Directory database, so the zone database would be replicated to only domain controllers of that domain.

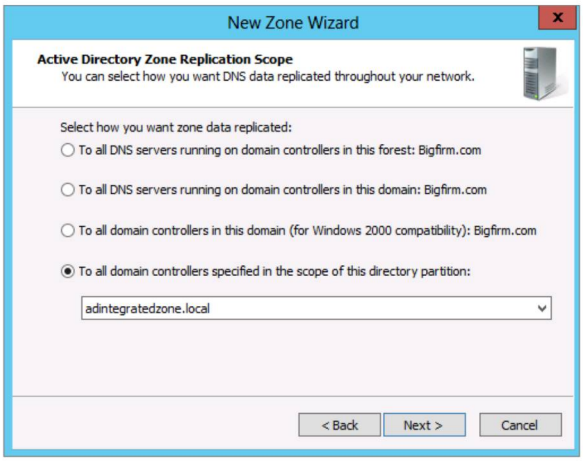


FIGURE 6.18
Active Directory Zone Replication Scope screen

If you want to create custom application partitions, such as the one displayed in the last option in Figure 6.18, then you will need to make use of either the `DNSCmd` utility or the `Add-DNSServerDirectoryPartition` cmdlet in PowerShell. Both of these come built into Windows Server 2012 R2, and they provide the ability to manage these types of custom partitions. From a PowerShell session with administrator permissions, type the following command to create a new application partition for the Active Directory integrated zone listed earlier. The name of the partition doesn't have to match the zone name; however, using the same name makes it more understandable when viewing the configurations:

```
C:\Users\administrator.BIGFIRM>Add-DNSServerDirectoryPartition -Name
"adintegratedzone.local"
```

Once the new partition has been created, an Active Directory integrated zone can be assigned to it, as shown in Figure 6.18.

Then the other domain controllers can be configured to support the zone using PowerShell also. On EC1, which is a domain controller for the Ecoast.Bigfirm.com domain, let's say you want to add these two Active Directory zones:

- adintegratedzone.local, which is placed in its own application partition
- The reverse lookup zone for 192.168.0.0 subnet, which is placed in the forest shared partition

First, add the server to the Name Servers tab in the properties of the desired zones, similar to Figure 6.15 shown earlier.

Then, use the following cmdlet to list the available partitions on EC1:

```
C:\Users\administrator.BIGFIRM>Get-DNSServerDirectoryPartition
```

From the output, you can see there are four application "directory" partitions. The first one is the custom one created earlier, which the server has enlisted in sharing. The second is the domain application partition that is shared with all of the non-Windows 2000 domain controllers within the Bigfirm.com domain. The third is for the Ecoast.Bigfirm.com domain. The fourth is for all the domain controllers within the forest of domains. EC1 is already enlisted for sharing these two partitions.

Using Stub Zones to Integrate with Other DNS Servers

The *stub zone* is another improvement that was first introduced with Windows Server 2003. Its concept and functionality haven't changed in Windows Server 2012 R2, and essentially it's an additional method to integrate with other DNS servers. The stub zone lists only the name server for a given namespace. It holds no control over the zone, so it indicates only what server could support name resolution for the namespace. Like conditional forwarders, it provides a lateral communication to the authoritative DNS server. These zones can also be replicated between domain controllers.

The New Zone Wizard sets up the stub zone with the following parameters:

- Type of zone: Stub
- Optionally stored in Active Directory with the desired application partition
- Namespace of the zone, such as Apex.com
- DNS server that supports this namespace

Once the stub zone is created, you can view its contents, as displayed in Figure 6.19. It lists the Start of Authority record for the namespace, the name server record for the namespace, and a host record for the name server.

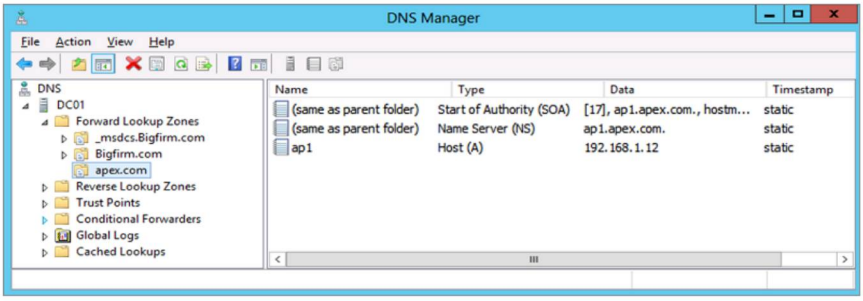


Figure 6.19 Stub zone example

Using Reverse Lookup Zones to Increase Security

You may have noticed that the zones we created were found in the Forward Lookup Zones folder within the DNS management console. A *forward lookup* means the client provides a fully qualified domain name and the DNS server returns an IP address. A *reverse lookup* does the opposite: the client provides an IP address, and then the DNS server returns an FQDN.

You might be wondering to yourself, "Why would this be necessary?" Well, the primary reasons are security related. Consider a hacker who has set up a malicious service to listen for DNS queries for FQDNs starting with www. on a network. When the rogue service gets a query, it automatically sends a bogus response to the client with the IP address of the hacker's web server. The website loads worms, viruses, Trojans, and other unsafe code before the user knows what's happening. Now, if the web browser could be configured to perform a reverse lookup on the provided IP address, it could compare the result with the queried name. If it didn't match, it wouldn't connect to the web server.

An example of how this type of reverse name resolution lookup works in the real world is with the SMTP service on Windows. This service has an option to perform reverse lookups on connections to the server. The SMTP servers provide their domain names in the communication, and the TCP/IP address is provided in the connection. The reverse lookup can then be performed to verify that things match up as they should do.

The `NsLookup` command illustrates the use of the reverse lookup. In the following code, the command is started in interactive mode to a server that doesn't have a pointer (PTR) record in a reverse lookup zone. Notice that the default server is listed as `UnKnown`. DNS queries can be flaky when this is the case.

```
C:\Users\Administrator.BF1>Nslookup
Default Server:  UnKnown
Address:  192.168.0.10
```

If the pointer record is created in the reverse lookup zone, the command output looks better. You can see here that it lists the name of the server:

```
C:\Users\Administrator.BF1>Nslookup
Default Server:  BF1.bigfirm.com
Address:  192.168.0.10
```

To correctly configure reverse lookup zones within your network, you need to understand how reverse resolution works. For IPv4, the IP address is in decimal dot notation with four octets, as in x.y.w.z. IPv6 is similar, but it uses hex numbers and a lot more. Either way, the process is the same. The DNS server that receives the query changes the order of the IP address. So, a query for the FQDN for the IP address of x.y.w.z becomes z.w.y.x, with .in-addr.arpa appended to the end of it. Then the DNS server attempts to resolve the FQDN of z.w.y.x.in-addr.arpa like a normal FQDN. It starts at the top-level domain of .arpa and works its way down to the in-addr. name servers. Each of the decimal values becomes a subdomain of the namespace to the right of it.

In small environments that include only one subnet, the subnet can be represented by a single zone. In our example, the 192.168.0.0 subnet is one zone, as shown in Figure 6.20. When the reverse lookup zone is created, the New Zone Wizard requests the name of the subnet to create the zone.

In larger environments where multiple subnets are in place, the zone for the higher-precedence octet needs to be created, and lower octets should be represented as subdomains or delegated subdomains. For example, if a large organization is using the 10.0.0.0 private IP addressing scheme, it would create a reverse lookup zone for the 10.in-addr.arpa domain name.

With dynamic updates occurring, subdomains would be automatically created for the next octet from 1 to 254, and pointer records would populate the subfolders of the structure. At some point, the subdomains could be delegated to another set of DNS servers, such as

domain controllers located in a site that contains these subnets. Then pointer records would be registered in the respective zone that represents the subnet.

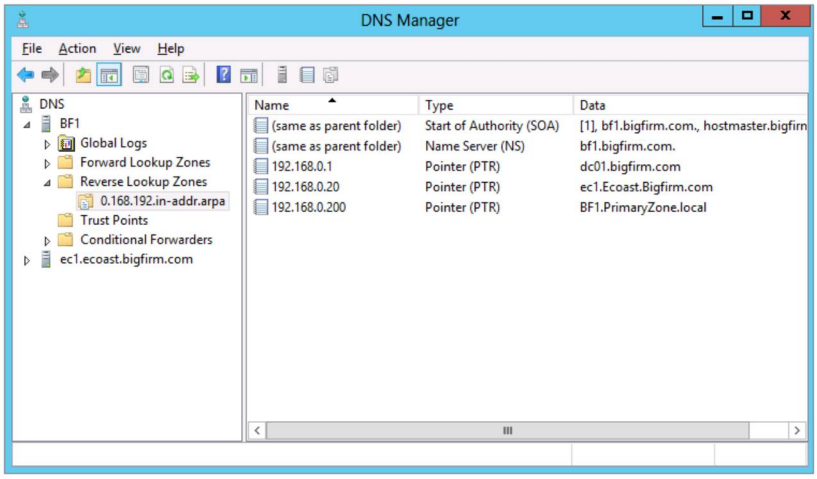


FIGURE 6.20
Reverse lookup zone example

In Figure 6.21, the 10.in-addr.arpa zone was created on BF1. If you wanted the 10.11.0.0 subnets controlled by another server, you would delegate the 11 subdomain. This was delegated to EC1. Within it, the actual subnet of 10.11.12.0 is also represented as a delegated subdomain.

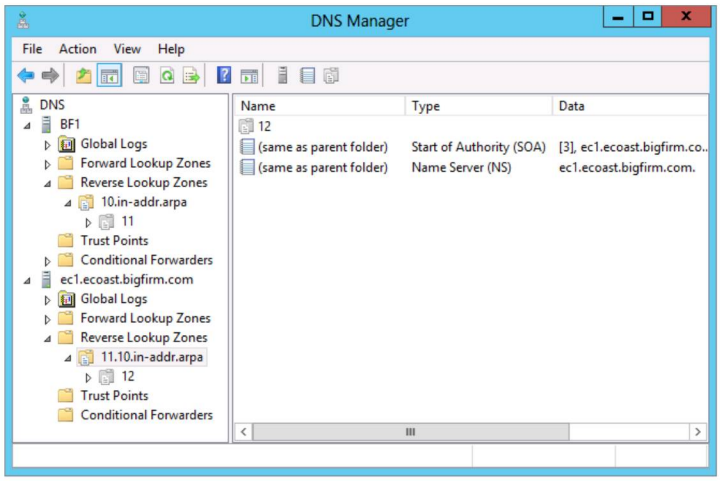


FIGURE 6.21
Reverse lookup zone for the 10.0.0.0 network

Understanding Record Types

Now that the databases are set up for the clients to retrieve information, you need to add records to the databases. As mentioned previously, Dynamic DNS (DDNS) is a process that allows DNS clients to register their hostnames in an assigned namespace, such as DHCP, and this adds the records for the Windows computers within an environment. However, it will still be necessary

to add some records manually and verify that the correct records were created dynamically. More than 25 record types are available for a DNS zone. In this section, we'll review the most common record types found in a Windows DNS implementation.

Round-Robin and Netmask Ordering

On the Advanced tab in the properties of the DNS server, you will find a number of server options enabled, two of which are:

- "Enable round robin"
- "Enable netmask ordering"

Round-robin This is a "poor man's" network load balancing technique. If you register multiple host records of the same name with different IP addresses, the DNS server responds sequentially with a different IP address for each query, starting with the lowest IP address. Although it doesn't spread the client load evenly or smartly to the available hosts, it still provides a balancing capability between servers.

Netmask Ordering Like round-robin, this uses multiple host records of the same name with a different IP address. Rather than picking randomly, the record that mathematically shows up as being closer is chosen. This is done through a comparison of the subnets. This is good if you have geographically separated hosts and your client needs to contact the one in its own network.

So, when using servers that are geographically separated, you need to decide which method is best. Netmask ordering will keep the response time minimized by nature of a shorter route. Round-robin will distribute the load more evenly if the clients are in a concentrated location.

However, a twist in these processes has come about when using Windows Server 2012 R2 with Windows 7 or Windows 8. The TCP/IP stack for IPv6 and IPv4 "when possible" will perform a similar process to netmask-ordering named, *default address selection*. Thus, it receives the IP addresses from the DNS server and decides on its own which is best.

Like most configurations, this can be overridden with a Group Policy object specifying the following registry change:

```
Hkey_Local_Machine\System\CurrentControlSet
\Services\Tcpip\Parameters\OverrideDefaultAddressSelection
```

A value of 1 will turn off default address selection and allow a random choice of the NLB round-robin servers.

You can see the effect on all this with the use of round robin for geographically split servers. For example, an environment with a disaster recovery site would like to offer two servers located in different sites to perform the same service. In this case, two FTP servers could be available to download data. DNS is rigged to provide two IP addresses for the same name, ftp.Bigfirm.com. Round-robin will resolve the records sequentially. If one server goes down, such as in a disaster, the clients will still have connectivity to ftp.Bigfirm.com. (It will occasionally select the dead server's IP address, but reconnects are still available to keep the data flowing.)

However, if netmask ordering or default address selection is in place, the clients may never get the live FTP server. The DNS server would be making the choice or the client would be. So, this gotcha scenario tells you to apply the time-tried rule: "Test, test, test." Validate which servers are being used in a normal scenario, and validate what happens when a disaster scenario occurs.

Host and Pointer Records

Host (A) and pointer (PTR) records are the most common records you will find in the forward lookup zones and the reverse lookup zones, respectively. The A records list the hostname of the computer and return the IP address. The PTR records list the IP address and return an FQDN.

You may have to create these for computers that do not have the DDNS update protocol available.

Alias Records

Alias (CNAME) records are created to list a secondary name for a computer. This record will list the name and return the assigned computer's FQDN. These are useful in the event of replacing the server of a published name that the clients use to access applications or services. Without their reconfiguration, clients will still be able to access the alias after replacement.

Mail Exchanger Records

Mail exchanger (MX) records are for SMTP communication. Mail servers request MX records to contact the receiving SMTP server for that namespace. Typically, you will be setting these up in an external DNS zone. However, they may be required internally for specific applications. The MX record requires an FQDN of the SMTP server and a priority value.

The priority helps determine which MX record to contact first, and next when more than one are available. The lower value wins out. Remember "number-one priority."

Say you have a primary SMTP server and a smart host SMTP server to support catching email when the primary server is unavailable. You would want to create MX records for both of them. The primary SMTP server needs a lower-priority value compared to the smart host, such as 10 and 20, respectively. When the primary SMTP server is unavailable, the smart host is contacted.

Service Records

Service records (SRV) are the "big kahunas" for Windows DNS implementations. Without the SRV records, workstations and servers would not be able to find domain controllers. The SRV records by themselves involve only five values:

Service Name This is a standard value typically preceded by an underscore, such as _gc. or _ladp. This is equivalent to a hostname and it would be tacked onto the FQDN of a service.

Server FQDN This is the server that provides the service.

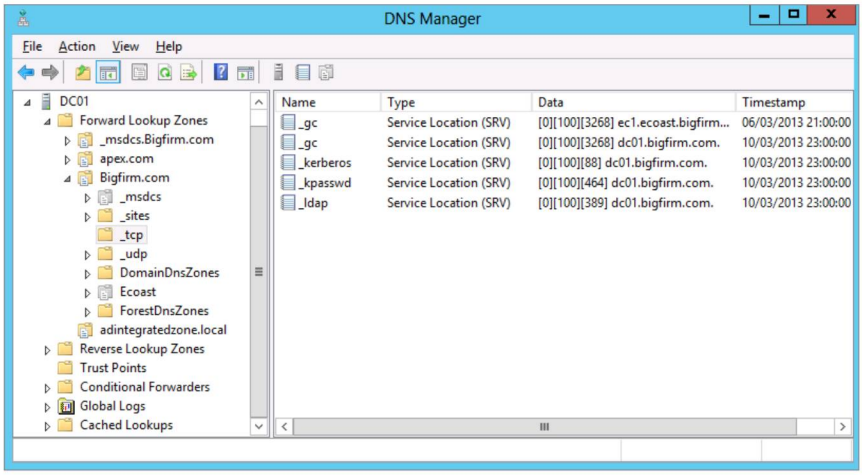
Port This is the TCP or UDP port on which the service is available. The protocol is signified in the registered name, such as _TCP.

Priority This works just like MX records—it has a "number-one priority."

Weight This is the tiebreaker for priority. Leave it at 0 if you're not concerned with ties.

You will see a plethora of SRV records in a Windows DNS zone that supports Active Directory. They are found in subdomain folders because the service name is assigned different FQDNs. The required service is described by the FQDN like _gc._tcp.bigfirm.com. Figure 6.22 illustrates the SRV records.

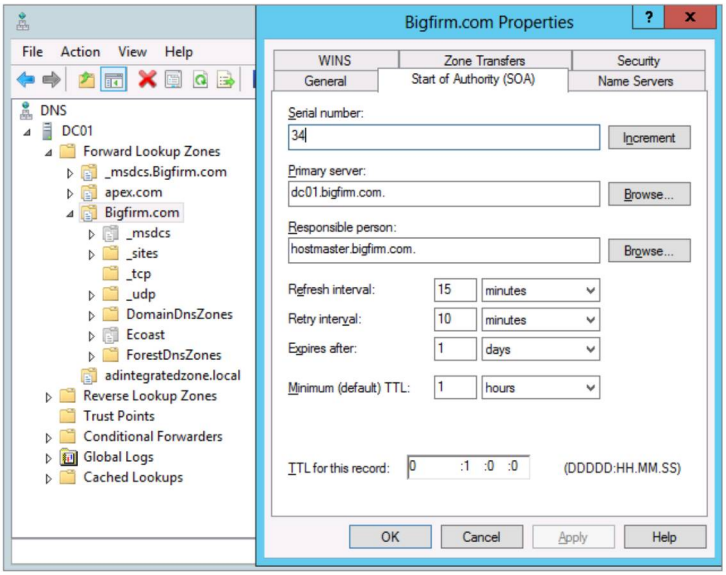
FIGURE 6.22
SRV record example



START OF AUTHORITY RECORDS

A Start of Authority (SOA) record is a single record within each zone. It has information about what DNS server controls this zone and parameters on how to treat the resolved records. It contains several values that shouldn't be modified by editing the record. You should edit them on the Start of Authority (SOA) tab in the properties of the zone, as shown in Figure 6.23.

FIGURE 6.23
Start of Authority (SOA) tab



The following are the fields on the tab:

Serial Number This is the revision number of the zone file. This really counts with standard primary zones because the Active Directory replication has its own serial number, so to speak. Secondary zones can compare their number with it to see whether their information is up to date. If it isn't, it's time for a zone transfer.

Primary Server This is the server in which the zone was initially set up. You can switch a primary server to a secondary server and vice versa if you want to modify how replication updates in your environment.

Responsible Person This is supposed to be an email address of the person who administers the zone. Notice that the @ is replaced by a dot (.). If you want to make someone really mad, you put their email address here!

Refresh Interval This is how long the secondary server can wait before attempting to check changes on the primary server. At this point of time, it compares the serial number of the SOA record with its own. By default it is 15 minutes. The value is listed in seconds within the actual record.

Retry Interval How long should the secondary server wait before trying again after a failed zone transfer? This is in seconds too and defaults to 10 minutes.

Expires After How long can the secondary server continue to answer requests on this zone after a zone transfer is performed? The default is one day. This is also in listed in seconds within the record, and the value is 86,400.

Minimum (Default) TTL How long should these records be cached or, in other words, what's the Time To Live (TTL)? The default is an hour, or 3,600 seconds.

Name Server Records

Name server (NS) records list the servers that can respond to queries for this zone. There will be at least one of these records in the zone. Like the SOA record, this is modified in the properties of the zone on the Name Servers tab, as shown earlier in Figure 6.15. The only value required in the NS record is the FQDN of the server. You'll notice a little note at the bottom of the Name Servers tab stating that the IP address is a retrieved value.

Managing DNS Clients and Name Resolution

You can deduce that every computer is a DNS client. The DNS service is a vital component of the network even if Active Directory is not part of it. In addition, it is the only method for getting to your favorite Internet websites like: `www.Sybex.com`.

On the Windows operating system, there are two areas concerning the clients of DNS: resolving hostnames and registering hostnames and IP addresses through Dynamic DNS updates.

Hostname Resolution

The Windows computer has two parts in the name-resolution process. This process is so important that we'll call it the "circle of life." One part, which is nearly dead, is NetBIOS, and the other is DNS. (You could call it the hostname process, but most admins call it DNS.) This circle consists of the steps a computer would take in resolving a given name, as illustrated in Figure 6.24.

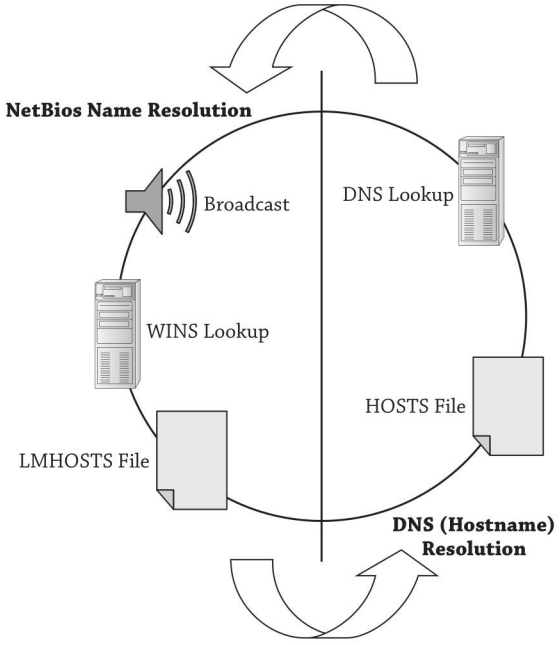


FIGURE 6.24
The circle of life

The NetBIOS process involves the following steps:

1. Broadcast the name into the network and see whether someone answers.
2. Look the name up in WINS.
3. Look the name up in the LMHOSTS file. This is another text file similar to the HOSTS file located in the same place: `c:\windows\system32\drivers\etc`. It lists NetBIOS names instead of hostnames.

The order of the first two steps is configurable, particularly through the DHCP server. The broadcast step can be skipped, or the WINS lookup can be skipped. The default on Windows Server 2012 R2 is to resolve through WINS first and then resolve with a broadcast second. However, the LMHOSTS lookup is always last.

The DNS process involves a shorter list of steps:

1. Look the name up in HOSTS.
2. Look the name up in DNS.

The order of the DNS process steps is not configurable, but you can modify the behavior of the DNS lookup. There are some good and bad points about the HOSTS file being first. If you are unable to access a DNS server or you need to redirect resolution of a name to another place, editing the HOSTS file works great. If the HOSTS file has stale or malicious entries, troubleshooting DNS can be difficult.

The name-resolution process circles through both parts until it comes up with an IP address. It also has a choice of where to start in the circle, either NetBIOS or DNS. This is basically

application dependent. Older legacy Windows applications looked at a name and considered it NetBIOS. TCP/IP-based applications thought it was a hostname. This affects how the name is resolved, and it still is part of the Windows operating systems.

Examples of this include the `net view` command and the `ping` command.

The `net view` command is a command from the ancient days of LAN Manager, which relied entirely on NetBIOS. If you attempt to connect to a server using this command, you will see the NetBIOS name cache populated with the server name. This is displayed with the `nbtstat -c` command. The cache can be cleared with `nbtstat -R`.

```
rem view the NetBios cache
C:\Users\Administrator.BF1>nbtstat -c

Local Area Connection:
Node IpAddress: [192.168.0.10] Scope Id: []

    No names in cache

rem access the shares on bfsc1
C:\Users\Administrator.BF1>net view \\bfsc1
Shared resources at \\bfsc1



Share name  Type  Used as  Comment

-------------------------------------------------------------------------------
NETLOGON    Disk           Logon server share
SALES       Disk
SYSVOL      Disk           Logon server share
Users       Disk
The command completed successfully.

rem view cache again
C:\Users\Administrator.BF1>nbtstat -c

Local Area Connection:
Node IpAddress: [192.168.0.10] Scope Id: []

                NetBIOS Remote Cache Name Table

        Name              Type       Host Address    Life [sec]
        --------------------------------------------------------
        BFSC1    <00>  UNIQUE          192.168.0.11      600
```

When you ping a server, the DNS process is used because it was written as a TCP/IP utility. You can see that it resolved the server via DNS by displaying the DNS cache with the `ipconfig /displaydns` command. The cache is cleared with `ipconfig /flushdns`.

```
rem clear the DNS cache
C:\Users\Administrator.BF1>ipconfig /flushdns

Windows IP Configuration

Successfully flushed the DNS Resolver Cache.

C:\Users\Administrator.BF1>ping BFSC1

Pinging BFSC1.bigfirm.com [192.168.0.11] with 32 bytes of data:
Reply from 192.168.0.11: bytes=32 time<1ms TTL=128
Reply from 192.168.0.11: bytes=32 time<1ms TTL=128
Reply from 192.168.0.11: bytes=32 time<1ms TTL=128
Reply from 192.168.0.11: bytes=32 time<1ms TTL=128

Ping statistics for 192.168. 0.11:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\Users\Administrator.BF1>ipconfig /displaydns

Windows IP Configuration

    BFSC1
    ----------------------------------------
    Record Name . . . . . : BFSC1.bigfirm.com
    Record Type . . . . . : 1
    Time To Live  . . . . : 1185
    Data Length . . . . . : 4
    Section . . . . . . . : Answer
    A (Host) Record . . . : 192.168. 0.11
```

Knowing this helps you decide how to support the DNS client-resolution process and affectively kill or support (as needed) NetBIOS names. The NetBIOS name process is chatty and takes up unnecessary CPU cycles. If the DNS server and client implementation are configured correctly, NetBIOS name-resolution support can be eliminated or at least minimized.

Configuring Clients

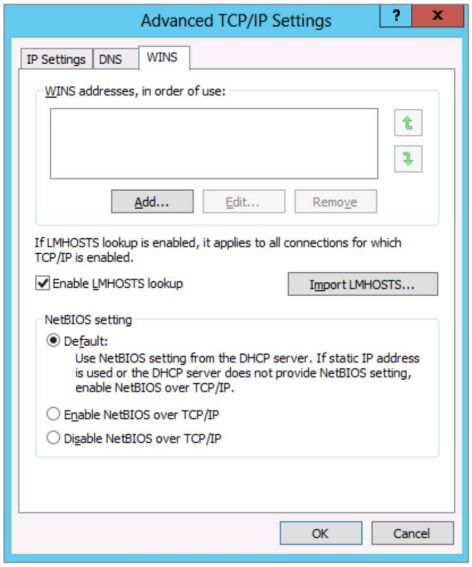
You can find the DNS and NetBIOS configurations in the IP properties of the network connection. Both are in the advanced settings. You can find the NetBIOS configurations on the WINS tab. Figure 6.25 displays the WINS tab with the default configurations.

The settings enable LMHOSTS and default the NetBIOS configurations to the DHCP server settings. First, by default the LMHOSTS file is empty. It would be good to disable this because malicious viruses have entered data into this file in the past.

Next, the NetBIOS settings default to what's on the DHCP server. The DHCP server scopes have an option named NBT Node Type (046). This setting orders the first two steps in the NetBIOS process in the circle of life. Four options are indicated by a decimal value:

- Broadcast only: "b-node," 1
- Contact WINS only: "p-node," 2
- Broadcast first and then contact WINS: "m-node," 4
- Contact WINS and then broadcast: "h-node," 8

FIGURE 6.25
The WINS tab



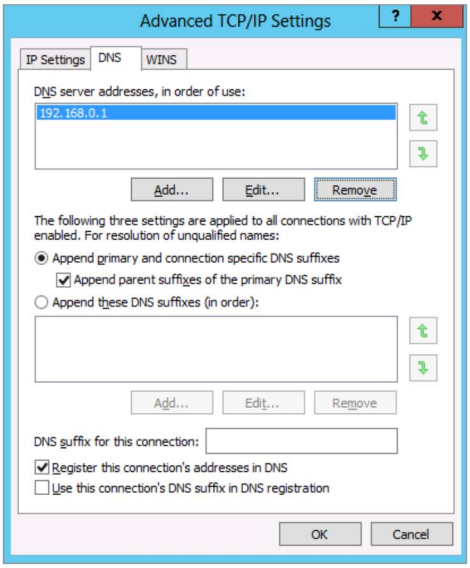
H-node is best for a network relying on NetBIOS because it reduces the chatter in a WINS-supported environment. If no WINS server is available, the computer can at least get some answer in a subnet, such as at home or in a workgroup. If DHCP is not configured with this value, the operating system's default configuration takes place. Windows Server 2012 R2 defaults to h-node, or hybrid mode.

It would be best for Active Directory environments to disable NetBIOS because it reduces the extra chatter and processes. It also helps reduce security threats such as bots that search networks for computers to attack using this naming system. Before doing this, however, you'd need to ensure that DNS name resolution is airtight.

You can find DNS client configurations on the DNS tab of the network properties, which is displayed in Figure 6.26. The obvious need is the DNS server's IP address. It should be connecting to the nearest server, typically a domain controller in its local site. A secondary DNS server is recommended.

The middle portion of the tab deals with unqualified names. This is a hostname without its "last name," the DNS suffix (such as BFSC1 listed earlier in the `ping` example). The DNS server needs an FQDN, so the DNS client appends DNS suffixes, the "last name," before sending the request. The primary DNS suffix is listed in the System control panel (My Computer properties) under the Computer Name tab. This is managed automatically by the operating system when the computer joins the domain, so you shouldn't need to mess with it. Additional suffixes may be necessary in larger environments, but in single-domain environments, the default settings work. Only in rare occasions would you consider adding a connection DNS suffix.

FIGURE 6.26
DNS tab



All of this is moot if FQDNs are used regularly. When applications are configured, use FQDNs of servers. When configuring home folders or folder redirection, use FQDNs. When writing scripts that map network drives, use FQDNs. Get the point? On top of this, using FQDNs bypasses the NetBIOS process. Applications can tell the difference between NetBIOS names and FQDNs and will resort to DNS when they recognize an FQDN.

Updating DNS Dynamically

To make the DNS name-resolution process airtight, you need to have all of the computers listed in DNS zones. In the past, DNS became a full-time job for system admins because they had to manually enter the ever-growing number of records for their network. To reduce the work for system admins, Microsoft sought a dynamic solution through WINS when developing NT and then switched to DNS when the Dynamic DNS (DDNS) update protocol was available. The process is pretty simple:

1. The client queries the SOA record for its primary DNS suffix namespace. This will tell it what server can accept DDNS. It also does this for the reverse lookup zone that its IP address is associated with.
2. The client makes the DDNS request to that server.

On standard zones' Start of Authority records, the primary server is listed. On Active Directory integrated zones, the domain controller receiving the request modifies the SOA response with its name. Since it can change the Active Directory database, there's no need to hunt down a domain controller located elsewhere. If the update process fails, it can attempt to find other name servers to perform the update.

The only configuration on the DNS tab concerning DDNS you can make is to the two check boxes on the bottom, as shown in Figure 6.26. You can register the name with the primary DNS suffix or with the connection suffix. The latter check box is deselected by default.

The silly part of this is that the DNS client service doesn't perform the DDNS process. The DHCP client service does. This reminds us of an eventful day when one of us disabled the DHCP client service on a domain controller. "This doesn't need the service running; it has a static IP address," he naively thought. As already mentioned, the DDNS process is used by the domain controllers to list the SRV records for Active Directory. Eventually, the clients couldn't find the domain controller because there were no SRV records for it. Oh, joy! Fortunately, this was in a lab environment.

There are two other locations that manage the DDNS process: the zone for a namespace and the DHCP server.

The DNS zone can be enabled for DDNS updates in the New Zone Wizard or can be modified in the properties of the zone. Figure 6.27 depicts the options for DDNS. The options are secure only, both secure and nonsecure, and disabled. Secure dynamic updates means the DNS client has authenticated with the domain controller prior to the update. Nonsecure means the update is accepted without authentication. Given the name, you can safely assume that hackers can exploit this option. Disabled prevents any DDNS updates from occurring.

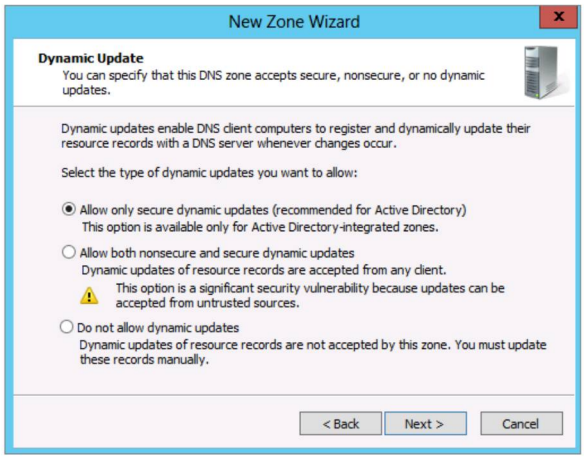


FIGURE 6.27 Dynamic DNS update options

The DHCP server can also participate in the DDNS process. In much earlier versions of Windows Server, there were plenty of Windows clients that didn't have DDNS capabilities. To resolve this, the DHCP server would identify these OSs and perform the updates for them. In addition, the DHCP server would perform the update upon request. Figure 6.28 shows the DNS tab of the IPv4 properties within the DHCP server.

The default settings are displayed, and these settings will be rarely modified. Essentially, DHCP is not performing any updates because clients are doing it themselves. It does perform a cleanup when leases expire. Name protection, which is just a check box reached by clicking the Configure button, basically prevents DHCP server from updating an existing DNS record.

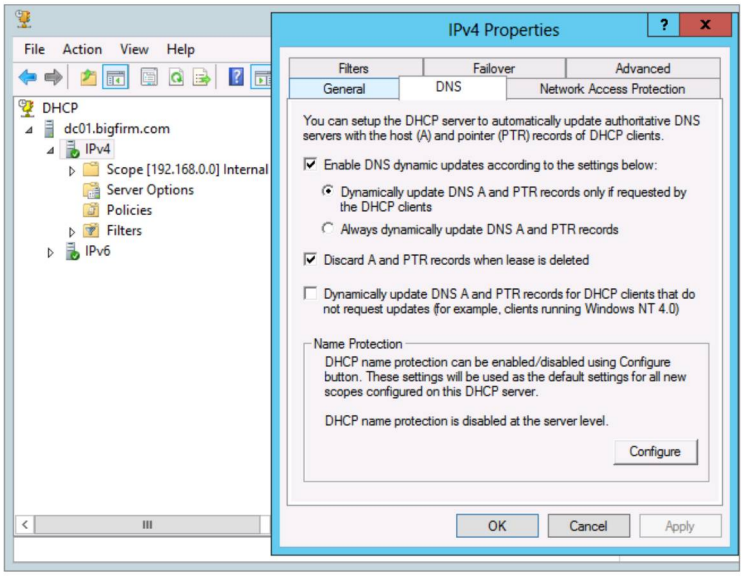


FIGURE 6.28 DDNS options on DHCP

Understanding Active Directory's DNS

Microsoft has integrated DNS and Active Directory so tightly together that it's difficult to discuss the two separately. When an Active Directory environment is created with Windows Server 2012 R2, the Active Directory installation process automatically configures DNS while adding the role. This provides a hands-off capability for the IT generalist in setting up DNS.

In the following sections, we'll cover the way Active Directory configures DNS and uses it to support clients. For a more detailed discussion of the Active Directory terms and concepts, refer to Chapter 7.

Configuring DNS Automatically

Windows Server 2012 R2 offers two ways of installing the DNS service: adding the DNS role on its own (as you've seen earlier in the stand-alone non-domain configuration) or adding the Active Directory Domain Services role. When you opt for the latter, you must run the Active Directory Domain Services Installation Wizard, which performs a number of configurations for the Active Directory Domain Services role, including automatically configuring and integrating the DNS role. In Chapter 7, we will walk you through the Active Directory installation process in detail, but here we take a look at just what happens regarding the DNS role.

First, you must understand the prerequisite for the Active Directory installation process. The prospective domain controller needs to have connectivity to the existing Active Directory DNS structure. Otherwise, it won't be able to connect to the domain controllers and obtain the necessary information. So, the IP configurations must list a DNS server within the Active Directory environment, preferably the forest root's DNS server or a DNS server in the same domain you are about to join. The only exception to this is when you create the very first

domain controller in the Active Directory environment. At that point, there is no Active Directory DNS structure to point to.

When the Active Directory Domain Services wizard is run, a new domain controller is configured. Depending on the options selected within the wizard, a new domain may be created. In either case, the DNS service and settings are configured automatically. The following changes are made.

Creating Application Partitions

Application partitions are divisions within the Active Directory database that are created for sharing DNS zones between different domains when a new domain or forest is created. The DomainDNSZones.domain.name partition is created for domain controllers within a domain. The ForestDNSZones.domain.name partition is created for sharing between domain controllers of an Active Directory forest.

If you look again at Figure 6.9 from earlier, you will notice the _msdcs.bigfirm.com subdomain is delegated just like Ecoast. It is delegated to the same domain controller, in this case, DC01.Bigfirm.com. The _msdcs.Bigfirm.com zone is created in the ForestDNSZone.Bigfirm.com application partition. This allows this portion of the namespace to be replicated to all domain controllers in the forest.

When additional domain controllers are created, they are automatically enlisted to these application partitions.

Adding a Forwarder

Within the DNS server's properties, a forwarder is added and can be viewed and configured from the Forwarders tab. This will typically be the IP address of the original DNS server that the server was using.

Modifying IP Properties

The new domain controller created by the Active Directory Domain Services Wizard is also a new DNS server. The primary DNS server's address in the IP properties is reconfigured to the loopback IP addresses, ::1 (for IPv6) and 127.0.0.1 (for IPv4).

Delegating the Subdomain

A child domain has a name that is a subdomain within an existing domain namespace. For example, Ecoast.Bigfirm.com is a subdomain within the Bigfirm.com namespace. When the Active Directory Domain Services Wizard does its thing, the new domain's namespace will be supported on the new domain controller as a delegated subdomain.

On the parent domain, a subdomain like Ecoast.Bigfirm.com is delegated to the new domain controller. The delegation will link the parent to the child domain for name resolution. This was illustrated earlier in Figure 6.9.

Additional Recommended Configurations

After a domain controller is created, we recommend that you make the following changes to the DNS settings:

1. In the network adapter TCP/IPv4 properties, change the primary DNS server to the IP address of the primary network connection. For example, if the primary IP address for your server is 192.168.0.1, then this is what you need to change your primary DNS server property to.

   When troubleshooting DNS with the `NsLookup` utility, the loopback address (127.0.0.1) causes tests to be "unauthoritative." Although this may be merely cosmetic, we have found results to be a little fickle using the loopback address, so it's best to stick with the primary IP address instead.

2. Create the reverse lookup zones in the ForestDNSZones.domain.name application partition.

   The reverse lookup zone for subnets may need to be shared between domain controllers of different domains.

3. Create a stub zone for new domain trees on the root DNS server.

   A domain tree has a different name than the root DNS server. Since the original DNS server entry is listed as a forwarder like the other domain controller promotions, the DNS server can communicate with the rest of the Active Directory DNS structure. However, there are no automatic configurations on the rest of the Active Directory DNS structure to resolve names in the new namespace. For example, we need to set up a conditional forwarder or a stub zone to point the DNS servers to the new domain controller for Apex.com. In Figure 6.19, you can see a stub zone used to assist resolving Apex.com domain FQDNs.

Understanding SRV Records and Clients

Looking at a brand-spanking-new domain's DNS zone, you will notice that it has a lot of new folders or subdomains. Drilling through these folders, you will find service location records by the boatload, as shown earlier in Figure 6.22. As we mentioned, Microsoft needed SRV records and Dynamic DNS updates to make Active Directory work. This is the result of the two technologies working together.

The `netlogon` service performs DDNS requests to create the SRV records within the Active Directory DNS namespace. The sole reason is to ensure computers can find domain controllers in the domain.

Within the Windows operating system processes, the specific services are sought out with the use of the DNS. In Figure 6.22, you will notice a few different services:

- `_gc`, or global catalog: The LDAP service to look up data within the global catalog
- `_kerberos`: The authentication process
- `_kpassword`: Another part of the authentication process
- `_ldap`: The LDAP service to look up data within the domain

Each of these services is performed by domain controllers within the domain or forest. In Figure 6.22, you see DC01.Bigfirm.com perform all of these roles and what TCP port it is listening on.

So, when a Windows computer needs a specific domain controller service, such as LDAP, it will request an SRV answer for _ldap._tcp.Bigfirm.com. It will then have all it needs to get busy with an IP address and port.

If a Windows computer needs to find a domain controller in its own site, it can look for it within the _sites.Bigfirm.com subdomain. This subdomain will list all the created sites within the Active Directory Sites and Services console.

The idea that admins could possibly support this load of DNS entries manually is incredible. For one domain controller, you can expect at least 16 to 20 different SRV records to be registered. Learning all of them is a daunting task for sure. That's when tools like `DcDiag` come into play. See the section, "Leveraging NsLookup and DcDiag," later in this chapter for instructions for using these utilities.

Windows Server 2012 R2 Additional Features

Up to this point, the essential features we have discussed and skills you have learned would go a long way toward ensuring you are proficient enough to manage a Windows Server 2012 R2 DNS environment. This section will discuss some of the additional DNS features in Windows Server 2012 R2 that tend not to be implemented as frequently as the other essential features that we have covered already, but nonetheless, they're definitely worth explaining here.

Global Query Block List

There are a few common host records that can be registered in DNS by other services. Web Proxy Automatic Discovery Protocol (WPAD) is a very common one. This helps web browsers automatically download the proxy configurations from a server. Since the record doesn't belong to a specific computer, any computer—including potentially compromised hacker computers—could attempt to register the name. Another common host record is Intra-site Automatic Tunneling Addressing Protocol (ISATAP). As you learned from Chapter 4, the purpose of ISATAP is to perform routing for IPv4 to IPv6 networks.

The global query block list specifies the names blocked from DDNS registration. Thus, a hacker computer's attempt to register names, such as WPAD or ISATAP, is rejected.

The following commands illustrate how you can administer this list. You can see this list with `Get-DNSServerGlobalQueryBlocklist`. Notice that it is populated with `wpad` and `isatap` by default:

```
C:\Users\Administrator.BF1>Get-DNSServerGlobalQueryBlocklist

Enable:  True
List:  {wpad, isatap}
```

If you want to add a name to the list, such as `www`, you can use the `Set-DNSServerGlobalQueryBlocklist` option. By default, the feature is enabled. You can disable or reenable it using the `-Enable` option with either a `$True` or `$False` Boolean value.

Global Names and Single Name Resolution

Even with the decline in the use of WINS, there seem to be some requirements in the marketplace to support some applications' use of the NetBIOS naming process. The GlobalNames feature is a special zone created to resolve a NetBIOS name (15 characters with no

dots in it). The DNS client is supposed to perform the query to the GlobalName zone when the primary and alternate DNS suffix searches have failed.

The steps to configure this are not difficult given the earlier discussions:

1. Create a new zone with the name GlobalNames.

   An Active Directory integrated zone is recommended to provide replication to other domain controllers.

2. Enable GlobalNames support with the `Set-DNSServerGlobalNameZone` command:

   ```
   C:\Users\Administrator.BF1>Set-DNSServerGlobalNameZone -Enable $True
   ```

3. Replicate the zone to other domain controllers.

   Remember to add these domain controllers to the name servers list of the zone.

4. Add CNAME records in the zone to redirect to specific hosts.

   In our example, `www` is redirected to hostrecord.PrimaryZone.local, as shown in Figure 6.29.

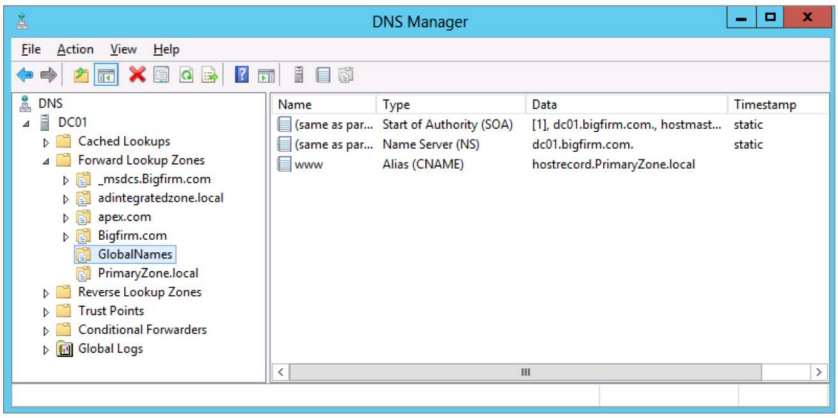


FIGURE 6.29
The GlobalNames zone

5. Add a service location record if you need other Active Directory forests to query this zone.

You can use the `NsLookup` utility to test the resolution of the global name:

```
C:\Users\Administrator.DC1>Nslookup
Default Server:  DC01.bigfirm.com
Address:  192.168.0.1

> www
Server:  DC01.bigfirm.com
Address:  192.168.0.1

Name:    hostrecord.primaryzone.local
Address:  192.168.0.21
Aliases:  www.bigfirm.com
```

In most environments that rely on Windows servers, the need for single names (NetBIOS) has been worked around, as we discussed earlier. It has also been minimized through the proper deployment of applications by using FQDNs and leveraging DNS.

Background Zone Loading

Some older environments have DNS zones so large that it takes the domain controllers more than an hour to restart the DNS service. If you have that problem, you're in luck with background zone loading! We expect a DNS zone must have a massive number of records to cause this issue.

While the DNS service is starting, it will start responding to zones it has loaded. Requests to zones that haven't loaded could be referred to other DNS servers.

DNSSEC

Like HTTP, DNS is an unencrypted and unauthenticated protocol. As we mentioned about reverse zones, hackers can spoof DNS responses. To counter this, the DNS Security Extensions (DNSSEC) were developed, which allow a DNS server to digitally sign the resource records. Windows Server 2012 R2 provides the support to act as secondary zones for a DNSSEC zone. It only responds to queries for the record from a digitally signed zone. It will also provide the necessary resource records to authenticate the signature.

These records are the KEY, SIG, and NXT records. KEY is the public key of the signing DNS server. SIG is the digital signature of the resource record. NXT basically lists all the valid records in the namespace.

With Windows Server 2012 R2, DNSSEC has the following enhancements on its predecessor:

- Active Directory integration and support for DNS dynamic updates
- Updated DNSSEC standards support (NSEC3 and RSA/SHA-2)
- Record validation through the use of the updated DNSSEC standards
- Additional PowerShell support for DNSSEC

If you want to test drive DNSSEC in your lab, then check out this step-by-step guide from Microsoft:

```
http://tinyurl.com/dnsseclab
```

Trust Anchors

Trust anchors are the public certificates of DNSSEC servers that the DNS server will trust for communications. The trust anchor certificates will be used to validate the digital signatures of the responses. These are added to the properties of the DNS server in the form of public keys.

Windows Server 2012 R2 contains the following new enhancements for trust anchors:

- Use of Active Directory for trust anchor distribution
- Automated rollover support
- Simplified extraction of the root trust anchor

Trust anchors are displayed in Windows Server 2012 R2 from the DNS Manager console, inside the Trust Points folder that you can see in the tree view on the left side.

Supporting Internet-based DNS Resolution

Within an organization, there is the need to manage Internet namespaces as well. The users within a LAN will need to access websites and other Internet-based services. External users will need to access the organization's websites and mail servers at a minimum. So, you will have to be mindful of these requirements also.

To allow external users access to your websites, an external DNS domain needs to be in place. Thus, you need to consider whether deploying an external DNS server is necessary. The internal computers will resolve external names through the internal DNS servers. Therefore, integration with the Internet DNS structure is required.

Supporting External DNS Domains

Most companies register a DNS namespace to support a website and email. Small- and some medium-size companies will allow an ISP to manage the namespace on their external DNS servers. The benefits are the availability of the servers and the reduced headache of maintaining additional servers on a public Internet-facing subnet. These servers are managed through a web interface and allow only a few types of records such as host, `cname`, and MX.

Using a Windows Server 2012 R2 server for external DNS operations is possible. The DNS role can be installed on a server that isn't a domain member (as we discussed earlier in "Configuring a Stand-Alone DNS Server"), and then the registered DNS namespace's name server record can be modified to the public IP address of the server. Oh, sure, exposing a stand-alone DNS server to the Internet in this way can be done, but realistically there are some cons to the idea:

- Windows Server 2012 R2 isn't free, and utilizing it simply as an external DNS solution is not the most economical way to do things.
- The Windows Server 2012 R2 server needs to be locked down and secure if exposed externally. Now you are looking at deploying Server Core and hardening that.
- It needs to be highly available, so you'll have to cluster the server or set up multiple DNS servers.
- If it hasn't already been taken care of, you'll need to have a highly available network connection to the Internet, too.

The cost of going in this direction is high, and many companies would rather spend the money elsewhere. This may be where a simple Linux implementation of DNS has an advantage. However, we recommend the approach most have taken—letting an ISP manage the namespace.

Split Brain

Many companies have also implemented a "split-brain" scenario when it comes to DNS, although not intentionally. What this means is they have an internal namespace that is the same as the external namespace. For example, we have the registered external namespace Bigfirm .com. The company has decided to build an Active Directory environment with the same name.

Caveats to Using Split-Brain DNS

Depending on whom you speak to or what technical document on DNS you read, you might see recommendations telling you not to use the same domain name for your company's internal and external DNS namespace. They will recommend that you use something different that may not be found on the Internet or a registered name you don't ever use.

When companies don't follow this recommendation, they soon realize that they have a conflict with resolving external resources that they own such as www.Bigfirm.com. The internal DNS server can't find the name, so it kicks back a big goose egg for a response. The admins try to fix this by adding the name manually with the external IP address, but adding the external IP address causes routing issues. In addition, the developers are whining that they can't upload new content with the external IP address. They need the internal IP address. This kind of additional administration hassle becomes the norm for dealing with split-brain issues.

Managing this scenario with just one server would be ideal. A split-brain DNS implementation is a cool idea that is supposed to remedy this issue. You could have a single DNS server support an internal and external zone of the same namespace. The IP addresses for the external zone would be provided to external requests and internal IP addresses for the internal requests.

It's a nice idea that Windows Server 2012 R2 doesn't support. Primarily, Microsoft doesn't support your organization exposing the domain controller that hosts the internal DNS namespace to even the edge of the Internet. This is not a secure measure. The Active Directory database is too valuable to hang in a DMZ like a ripe peach. So, you have to come up with an alternative.

Your objective is to provide resolution of external requests with external IP addresses and internal requests with internal IP addresses. Using Microsoft DNS, you will have to administer (a minimum of) two DNS servers to support this scenario. Here are the basic steps:

1. Implement an external DNS server to support Bigfirm.com. Typically, this is already in place with the registration of a domain name with the help of the ISP.
2. Implement the internal DNS structure. Using the Active Directory Domain Services Installation Wizard, this is handled quite readily.
3. Add any external records to the internal zone for Bigfirm.com.

   Remember, the DNS servers within the network will be authoritative for the Bigfirm.com domain. If it can't find www.Bigfirm.com, it doesn't exist. External records must be duplicated in the internal zone so a positive result can be returned. You will have to test routing to ensure that the IP address is accessible. If routing causes problems, you may have to use an internal address.
4. Configure resolving external namespaces using root hints or forwarders. This topic is covered in the following section.

Resolving External Namespaces

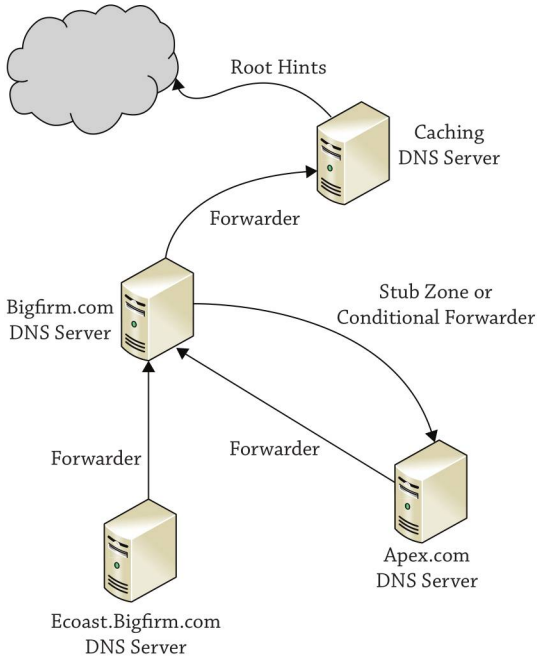
We discussed how to integrate a DNS server with others. The primary methods of resolving DNS names in the Internet are the root hints or the forwarders. The root hints were a list of DNS servers that were at the top of the Internet's DNS structure. The DNS server could communicate

with these servers to perform recursive queries for external namespaces. Forwarders were lateral requests to another DNS server to see whether that server could come up with the name. We mentioned that in small environments we prefer to use forwarders to an external DNS server that is supported by the ISP, but using root hints still work in this scenario too.

It is important to not mix the two. Don't list root hints servers as forwarders. A query to a root hint is a referral request that always returns the name server for a domain. It doesn't respond with host records, and it doesn't perform the recursive operation that a forwarder would. Forwarders also take precedence over root hints. In Figure 6.10 shown earlier in the chapter, the Forwarders tab includes the check box "Use root hints if no forwarders are available." You can infer that if a forwarder is listed and it comes up with an invalid response, the query is over, and root hints won't be touched.

In extensive internal DNS environments, judicious use of the forwarders and root hints is necessary. The internal subdomain name servers need to resolve queries from the root DNS server. They also have to resolve Internet-based queries. Taking advantage of the caching capability of DNS, they can rely on a server to resolve and store common queries to reduce externally bound traffic. Microsoft recommends that the caching server should not be the root server so the root server will not be overburdened with the additional workload. In addition, they warn internal DNS servers that host zones should not communicate directly with the Internet to reduce their exposure to Internet. Figure 6.30 depicts one solution that could work for our fictitious DNS structure.

FIGURE 6.30
Internal DNS structure



In this example, forwarders are used to send requests to the root DNS server in Bigfirm .com. Root hints could be used by removing the Internet root hints and listing BF1.Bigfirm.com as the root hint server. The caching server is in place to prevent the DNS servers hosting the

Active Directory integrated zones from making queries into the Internet. It performs resolutions via the root hints. To handle queries in the Apex.com domain, a stub zone or a conditional forwarder is used.

Other solutions are possible and may have merit for different reasons. We prefer the simplicity of using the forwarders to manage the integration between the servers.

Administration and Troubleshooting with DNS Tools

In this section, we'll discuss the available tools and troubleshooting techniques for DNS name resolution. Given the importance of DNS, you need to be familiar with the tools that give valuable information to discern where problems lie with name resolution. The standard admin tools, the DNS management console, and PowerShell provide additional information over and above configurations. The `NsLookup`, `DcDiag`, and `DNSLint` utilities provide excellent initial indications of problems concerning DNS resolution.

Administering the DNS Server with the DNS Management Console and PowerShell

To administer the DNS server, we have touched on two tools: the DNS management console, which is an MMC snap-in, and PowerShell, which, of course, is a shell-based command-line tool. PowerShell offers the capability of administering the entire server like the MMC and offers a little more functionality. For example, as you learned earlier, the DNS management console doesn't offer a method of modifying the global query block list or creating directory partitions.

Throughout the chapter, you have seen the DNS management console used to create zones and edit the properties of servers and zones, which are the run-of-the-mill types of tasks you do with it. You can also take advantage of some diagnostic configurations within the console. These are configured in the properties of the DNS server.

Event Logging A separate Event Viewer log is created for the DNS service. It is attached to the DNS management console. In addition, the server collects all events by default, which is set on the Event Logging tab.

Debug Logging More detailed logging of the actual communication occurring on the DNS server can be gathered for inspection. Debug logging is disabled by default but can be enabled from the properties of the DNS server; Figure 6.31 displays the Debug Logging tab. This feature is valuable if the DNS server is not acting reliably. Although most DNS issues are resolved with proper IP connectivity, you will find this tool useful when IP connectivity is determined not to be the cause. On rare occasions, we have had to verify whether specific requests were hitting the server, and this tool provided the information.

Monitoring This tab can also be accessed from inside the DNS server properties and is displayed in Figure 6.32. It's the equivalent to the million-dollar machine that goes "ping!" It's like a blinking light. It can test DNS queries from this server or to another server—not a specific server, mind you, just any indiscriminant server out there. Then you can perform the test at intervals.

The output is just pass or fail. Basically, if the DNS server is having problems, it's a fail. We haven't found any comfort or valuable data in resolving DNS issues on this tab, and most likely it won't tell you anything new. We'd recommend using something like Microsoft's System Center 2012 R2 Operations Manager for monitoring your DNS servers, and this topic is discussed later in Chapter 30.

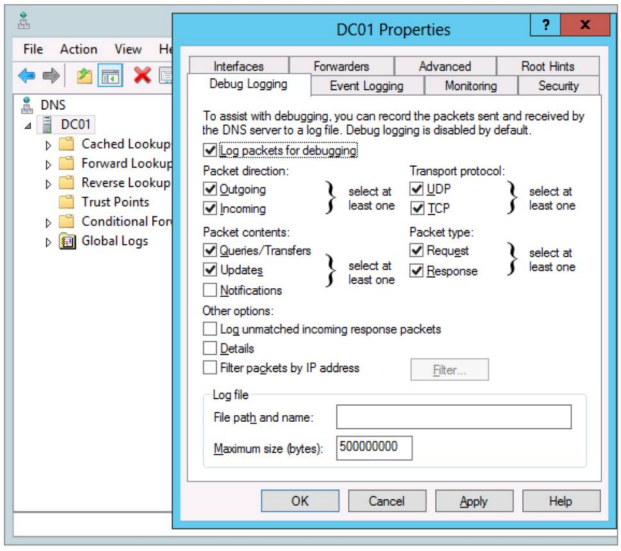


FIGURE 6.31
Debug Logging tab

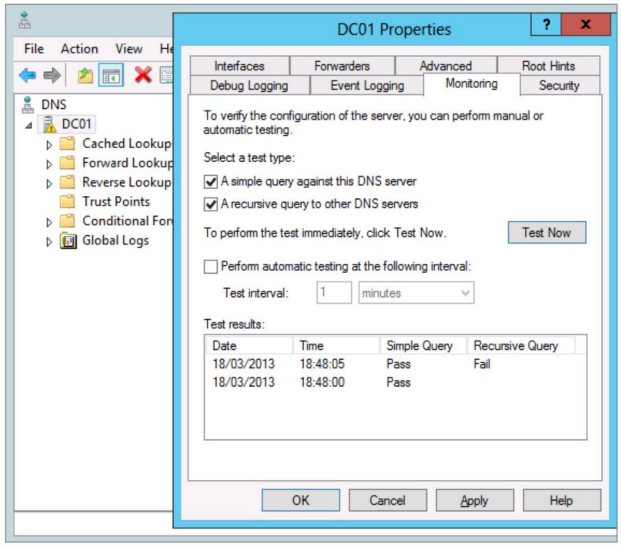


FIGURE 6.32
Monitoring tab

PowerShell offers a number of diagnostic cmdlets as well. These may be useful in collecting and examining data:

- `Get-DNSServer` provides configurations of the DNS server.
- `Get- Get-DnsServer | Export-Clixml -Path "c:\config\DnsServerConfig.xml"` generates a text file of the configurations and zone properties.

- `Get-DNSServerDiagnostics` provides event-logging details of specific DNS operations on the server.
- `Clear-DNSServerCache` empties the cache. Occasionally, stale resolved records need to be removed after a problem has been resolved. This is also available in the DNS management console.

Each of these tools provides administration and monitoring features. We find they are useful for the deeper investigation where the frontline tools such as `NsLookup`, `DcDiag`, and `DNSLint` do not immediately indicate the problem.

Leveraging NsLookup and DcDiag

`NsLookup`, `DcDiag`, and `DNSLint` are the tools that you will most likely use the most when troubleshooting DNS issues. `NsLookup` provides immediate indications of something that could be wrong or misconfigured with name resolution. `DcDiag` and `DNSLint` provide initial indications concerning Active Directory–related issues, such as DDNS registration and SRV records. If these tools do not reveal the problem, you can rely on the features of the DNS management console and PowerShell to help you out.

NsLookup

`NsLookup` is the first tool we go to when troubleshooting name-resolution problems. It connects to the listed primary DNS server in the IP configurations and makes requests for DNS queries.

Notice that the utility doesn't perform the name-resolution process, in other words, the circle of life. It narrows down to one portion of the circle. In our discussion on clients, the examples of the `ping` and `net view` commands were provided to show the different parts of the process. The `ping` example showed the DNS process that included the first step of looking the name up in the HOSTS file. So, you will see a disconnect between `ping` and `NsLookup` if the HOSTS file has records for the same hostname in it.

Conficker Virus

One example of malicious software that uses DNS to cause disruption is the Conficker virus. This "lovely" bit of code originated in 2008 and can (believe it or not) still be found today lingering on computers that haven't been patched with the latest virus definitions and system updates. It prevents browsers on infected systems from hitting important sites within specific DNS namespaces like Microsoft.com, Symantec.com, and Norton.com.

This has a crippling effect. The computer can't go to the Windows update site; it can't even look up possible solutions for the problem. Even if you installed Norton AntiVirus onto the machine, it couldn't access the Symantec update site for the latest definition updates. The machine couldn't fix itself!

When we came across Conficker, `NsLookup` was our first tool of choice to troubleshoot it. The queries to Microsoft.com and Symantec.com were positive, but it wouldn't fly in Internet Explorer or Firefox instances on that computer. This difference helped narrow down where the issue lied. Therefore, the browsers were infected.

Of course, NsLookup provides the IP addresses for the sites we need to hit. But—"Oh, no!"—the Microsoft websites don't play using just the IP address in the URL. So, trying that workaround doesn't work either.

We found the solution using the Microsoft Windows Malicious Software Removal Tool (MSRT). It had to be downloaded on a separate computer, and then using Sneakernet, we transferred the MSRT package to the infected computer. It identified and removed the virus. These days, thankfully, the Conficker virus is restricted to older operating systems with out-of-date or missing antivirus and Windows Update definitions. It's a good example, though, of how name-resolution modifications can be used to create havoc and then how tools, such as NsLookup, can be used to troubleshoot and identify its presence.

When an application cannot access a server, after checking TCP/IP connectivity, we start to poke around with the NsLookup command. Here are some things we look for:

- Is the DNS server responding? The command will tell you right at the start if it can connect to the DNS server. If there is a delay or time-out, there's no need to continue. You have a connectivity issue.
- Is the default server unknown? This indicates that the reverse lookup performed by NsLookup failed. When it is in this state, we find that the rest of NsLookup tests get squirrely.
- Can you resolve a local FQDN? This bypasses a portion of the client's processing. The client will append primary DNS suffixes to search for hostnames.
- Can you resolve a hostname without the DNS suffix? This is what the client does so you can walk through the step.
- Can you resolve external FQDNs? This will validate that the default DNS server can get out to the Internet.

NsLookup has two methods: single-command queries and interactive mode. The interactive mode is much more powerful, so we opt to use that at the start. This mode offers the ability to perform queries on different types of resource records and allows you to switch to another server. The following are examples of queries we commonly perform (with remarks added for clarity):

```
C:\Users\Administrator.BF1>Nslookup
Default Server:  BF1.bigfirm.com
Address:  192.168.0.10

rem a host record query
> BF1.bigfirm.com
Server:  BF1.bigfirm.com
Address:  192.168.0.10

Name:    BF1.bigfirm.com
```

```
Address:  192.168.0.10

rem a reverse “ptr” record query
> set q=ptr
> 192.168.0.10
Server:  BF1.bigfirm.com
Address:  192.168.0.10

10.1.168.192.in-addr.arpa     name = BF1.bigfirm.com

rem a start of authority query
> set q=soa
> bigfirm.com
Server:  BF1.bigfirm.com
Address:  192.168.0.10

bigfirm.com
        primary name server = BF1.bigfirm.com
        responsible mail addr = hostmaster.bigfirm.com
        serial  = 124
        refresh = 900 (15 mins)
        retry   = 600 (10 mins)
        expire  = 86400 (1 day)
        default TTL = 3600 (1 hour)
BF1.bigfirm.com      internet address = 192.168.0.10

rem a name server query
> set q=ns
> bigfirm.com
Server:  BF1.bigfirm.com
Address:  192.168.0.10

bigfirm.com     nameserver = BF1.bigfirm.com
BF1.bigfirm.com      internet address = 192.168.0.10

rem a resource record query
> set q=srv
> _ldap._tcp.bigfirm.com
Server:  BF1.bigfirm.com
Address:  192.168.0.10

_ldap._tcp.bigfirm.com  SRV service location:
          priority       = 0
          weight         = 100
          port           = 389
          svr hostname   = BF1.bigfirm.com
BF1.bigfirm.com      internet address = 192.168.0.10
```

DcDiag

DcDiag was originally part of the administrator support tools (which needed to be installed separately) from earlier versions of Windows Server but is now included by default as part of the Windows Server 2012 R2 installation. It's our first-choice tool to perform a quick health check on the DNS structure. Since it runs through a score of domain controller diagnostics, it must validate that DNS is working as needed. After running the standard battery of tests, you may see errors in attempting to connect to domain controllers. Then you could run additional DcDiag tests specifically for DNS. The following example tests whether a domain controller can perform DDNS to register the SRV records:

```
dcdiag /test:RegisterInDNS /DnsDomain:bigfirm.com /f:documents\
dcdiagRegisterInDNS.txt
```

This produced the following text output:

```
Starting test: RegisterInDNS

      DNS configuration is sufficient to allow this domain controller to

      dynamically register the domain controller Locator records in DNS.

      The DNS configuration is sufficient to allow this computer to dynamically

      register the A record corresponding to its DNS name.

      ..................... BF1 passed test RegisterInDNS
```

DcDiag performs a boatload of domain controller–related tests, including several DNS tests. We mentioned one, the RegisterInDNS test. These tests primarily focus on the integration between the DNS servers within an Active Directory environment. Tests can be performed on delegation, forwarders, dynamic update, and external DNS name resolution.

The following is a portion of the help information for the DcDiag utility. It lists the tests available for DNS. We normally rely on NsLookup to test external name resolution, so we never use the /DnsForwarders and /DnsResolveExtName tests but have listed them here for transparency:

```
DNS
This test checks the health of DNS settings for the whole enterprise. Sub tests
can be run individually using the switches below. By default, all tests except
external name resolution are run

/DnsBasic                (basic tests, can't be skipped)
/DnsForwarders           (forwarders and root hints tests)
/DnsDelegation           (delegations tests)
/DnsDynamicUpdate        (dynamic update tests)
/DnsRecordRegistration   (records registration tests)
/DnsResolveExtName       (external name resolution test)
/DnsAll                  (includes all tests above)
/DnsInternetName:         <internet name> (for test /DnsResolveExtName)
(default is www.microsoft.com)
```

As discussed earlier with SRV records, the number of SRV records registered by a domain controller is so great that it is difficult to eyeball whether it's working correctly. In addition to the `/registerinDNS` test, `/DnsDynamicUpdate` and `/DnsRecordRegistration` run through checks concerning SRV registration by the domain controllers. Unlike `/registerinDNS`, these do not have to be run locally on the domain controller. The following command will verify the SRV records for a domain controller. The `/v` option is for "verbose." The output is lengthy because it lists all of the SRV records for the domain controller:

```
C:\Users\Administrator.BF1>dcdiag /s:BF1.bigfirm.com /test:dns /
dnsrecordregistration /v
```

The following will validate that DDNS update is operational on a zone. It will register a host and delete it from the DNS zone of the server. In this case, that is Ecoast.Bigfirm.com:

```
C:\Users\Administrator.BF1>dcdiag /s:ec1.Ecoast.Bigfirm.com /test:dns /
dnsdynamicupdate /v
```

DIG Deeper than NsLookup

There's a DNS troubleshooting tool that's been around in the Unix world for quite some time called the Domain Information Groper, or DIG for short. If you ask users of DIG for their opinion on how the `NsLookup` utility stacks up against DIG as a DNS troubleshooting tool, don't be surprised if they start laughing before politely telling you it doesn't stack up at all!

The good news, though, is that you can download DIG for free (`http://www.isc.org/software/bind`) and install it into your Windows Server 2012 R2 environment to help turbocharge your DNS troubleshooting skills. DIG can be run in typical command-line format but also has a batch mode option that supports the reading of lookup requests from a file.

If you want to learn how to install DIG onto a Windows Server, have a look at the walk-through here: `http://tinyurl.com/DIGinstall`. You can also review an online usage guide for DIG from this link `http://tinyurl.com/DIGusage`.

Helpful DNS Troubleshooting Links

The tools we have discussed up to now have all been based on what's available out of the box with Windows Server 2012 R2. Those tools are where you should start your troubleshooting from inside the network. In this section, we want to share some DNS-related websites that you can use to assist with external DNS problems.

`www.IntoDNS.com` This is such a simple but effective website to add to your DNS troubleshooting armory. When you reach the home page, all you need to do is to enter the DNS name of the domain that you want to get information on and then click the Report button. This will then return all of the available information for the A (parent), NS, SOA, MX, and WWW records for that domain. With this information, you can quickly identify whether any of those records have been configured incorrectly, or you can just use it as a cross-check for the information that you have been given.

www.MXToolbox.com This site does pretty much what it says on the tin. It's designed to assist with MX record troubleshooting and can prove instrumental in trying to ascertain why mail flow for a particular email domain isn't working. You can run a number of different tests against any given domain, such as MX lookup, Blacklist check, Whois lookup, and SMTP verification. If you haven't used this site before, now's the time to add it to your favorites!

www.DNSStuff.com This is another popular site that you can use to run DNS reports, Whois, and IP information tests. You can also get access to a large number of additional DNS troubleshooting tools here if you're prepared to sign up for an account with them, but investigate the benefits for yourself with a trial run before purchasing.

The Bottom Line

Explain the fundamental components and processes of DNS. DNS relies on integrated servers that manage a hierarchical naming structure. On the Internet, this structure starts with root servers and then top-level domain servers, which delegate subdomains to other DNS servers. Within a DNS server, the database of records is known as a *zone*, and it can be replicated between other DNS servers to provide distributed query resolution for a given namespace.

> **Master It** Several common DNS records were discussed in this chapter. The SRV and MX records both have a parameter named `priority`. If there were two SRV records for the same service with a `priority` parameter of 10 and 20, which SRV record would be selected first?

Configure DNS to support an Active Directory environment. Active Directory requires a DNS namespace to be available to support the assigned name of the domain. Windows Server 2012 R2 provides an automatic capability to create the required DNS structure through the domain controller promotion process. The DNS zones can be stored in the Active Directory database, which provides multimaster replication of the DNS records. With the use of SRV records and DDNS update, the domain controllers can register their services in DNS for clients to access them.

> **Master It** The DNS service on DCs can create Active Directory integrated zones. In which locations within the Active Directory database can the zones be placed? What scope do these locations provide?

Manage and troubleshoot DNS resolution for both internal and external names. Internal and external name resolution relies on the connectivity between DNS servers. Forwarding and root hints are the primary methods to allow DNS servers to send queries between them. Several tools are available to assist troubleshooting and monitoring DNS configurations and performance, including `NsLookup`, PowerShell, and `DcDiag`.

> **Master It** The SRV record registration for domain controllers is performed by the `netlogon` service. It is a very complex and demanding task to attempt to perform this manually. What tests can be performed to verify whether SRV records are correctly registered within a domain?